Universidad Zaragoza
1542

Proyecto Fin de Carrera

ANEXOS
Desarrollo de un sistema informático de control y gestión de flujos de trabajo, Workflow Management System (WfMS)

Autor

David Roy Marquina

Director

Fernando Cortés Franco

Ponente

Santiago Velilla Marco

ESCUELA DE INGENIERÍA Y ARQUITECTURA

2017

endalia®
HR Consulting & Software

I. DOCUMENTO VISIÓN

WORKFLOW MANAGEMENT SYSTEM

DAVID ROY

TABLA DE CONTENIDOS

Tabla de contenidos 2
1. Introducción 3
1.1 Objetivo 3
1.2 Alcance 3
1.3 Organización del documento 3
2. Workflow 3
2.1 Definición 3
2.2 Componentes 4
2.3 Tipos 4
2.4 Comportamiento 5
3. Workflow Management System (WfMS) 6
3.1 Definición 6
3.2 Componentes 7
4. Glosario 7
5. Bibliografía 8

1. INTRODUCCIÓN

1.1 Objetivo

El objetivo de este documento consiste en conocer y estudiar los conceptos que van a ser manejados a lo largo de todo el desarrollo del proyecto, para así comprender mejor las necesidades y características del sistema a desarrollar.

1.2 Alcance

Este documento pretende mostrar una visión global de los principales componentes y funciones que servirán como punto de partida para las siguientes fases del desarrollo. Además, este documento pretende definir un glosario de términos que se usará a lo largo del proyecto y que permitirá identificar los distintos elementos de manera unívoca.

1.3 Organización del documento

El presente documento contiene las siguientes secciones:

1. Introducción: Descripción del objetivo y alcance del documento.
2. *Workflow*: Definición y componentes de un *workflow*.
3. Workflow Management System: Definición y componentes.
4. Glosario: términos consensuados que usaremos a lo largo del proyecto.
5. Bibliografía: Referencias a libros y webs utilizadas en el documento.

2. WORKFLOW

2.1 Definición

Al comenzar la investigación para este estudio se encuentran diferentes definiciones de *Workflow* disponibles en internet, de entre todas ellas se toma como referencia la que propone la *Workflow Management Coalition* [1], una organización mundial fundada en 1993 por desarrolladores, profesionales y grupos de investigación que tiene entre sus objetivos ser referencia a nivel global en la definición y desarrollo de estándares en el ámbito de los procesos de negocio. Su definición dice así:

"*La automatización de un proceso de negocio*[1]*, en su totalidad o en parte, en el que la* ***información*** *y* ***tareas*** *pasan de un* ***participante*** *a otro siguiendo unas* ***reglas*** *establecidas*". [2]

[1] Proceso de negocio: Un grupo de actividades relacionadas entre sí que se ejecutan conjuntamente para alcanzar un objetivo empresarial dentro del contexto de una estructura organizativa. Se diferencia de un workflow en que un proceso de negocio puede representar cualquier tipo de actividad de la empresa, mientras que un workflow representa únicamente aquellas actividades o partes de éstas que pueden automatizarse.

Un *workflow* representa un proceso en el que la información se va transmitiendo y transformando a lo largo de unos pasos siguiendo unas reglas. Se tiene por un lado una parte fija que define las reglas y comportamiento del *workflow* (**plantilla**) y por otro lado una parte dinámica que representa los procesos o ejecuciones que están en marcha (**instancias**), con su información particular y sus estados. Un *workflow* tiene una plantilla y puede tener cero o más instancias:

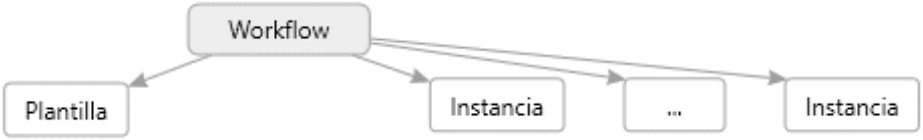


Figura 1: Esquema de las instancias de un workflow

2.2 Componentes

Con todo lo anterior se puede inferir los componentes de un *workflow*:

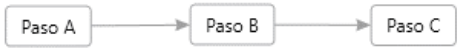
- **Plantilla**: define las reglas y el comportamiento del *workflow*.
  - Pasos: cada una de las fases del proceso. Éstas se componen de:
    - Responsables: los posibles **participantes** del paso.
    - Acciones: las **tareas** que se realizan.
    - Notificaciones: las notificaciones enviadas a los participantes.
  - Secuenciación: las **reglas** que definen el orden de ejecución de los pasos.
- **Instancia**: contiene la información de cada ejecución del *workflow*:
  - Estado: si el proceso está en curso o finalizado.
  - Campos: la **información** que se transmite está compuesta por campos de información.
  - Autor: el participante que ha ejecutado el paso.

Figura 2: Esquema de los componentes de un workflow

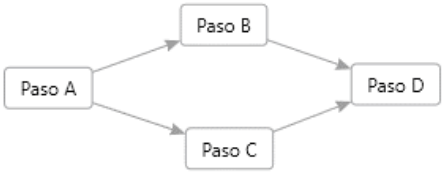
2.3 Tipos

Atendiendo a la secuenciación de un *workflow* se descubre un factor importante que se debe tener presente a la hora de desarrollar el proyecto. Según sea la secuenciación de los pasos se pueden clasificar los *workflows* en 2 tipos:

- **Lineales**: en todo momento sólo hay 1 fase en ejecución:



- **No lineales**: puede haber varias fases en marcha a la vez:



2.4 Comportamiento

Atendiendo al comportamiento de un *workflow*, éste puede realizar 2 operaciones fundamentales:

- **Avanzar** de un paso al siguiente (o siguientes como hemos visto en el apartado anterior).
- **Retroceder** de un paso al anterior (o anteriores).

Tomando como ejemplo los *workflows* del apartado 2.3, estos serían sus diagramas de estados:

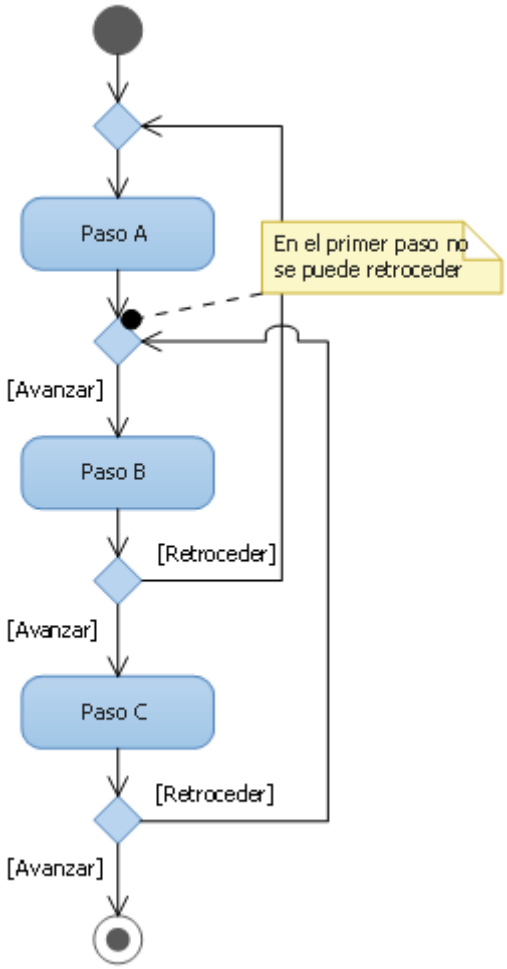


Figura 3: Ejemplo de workflow lineal

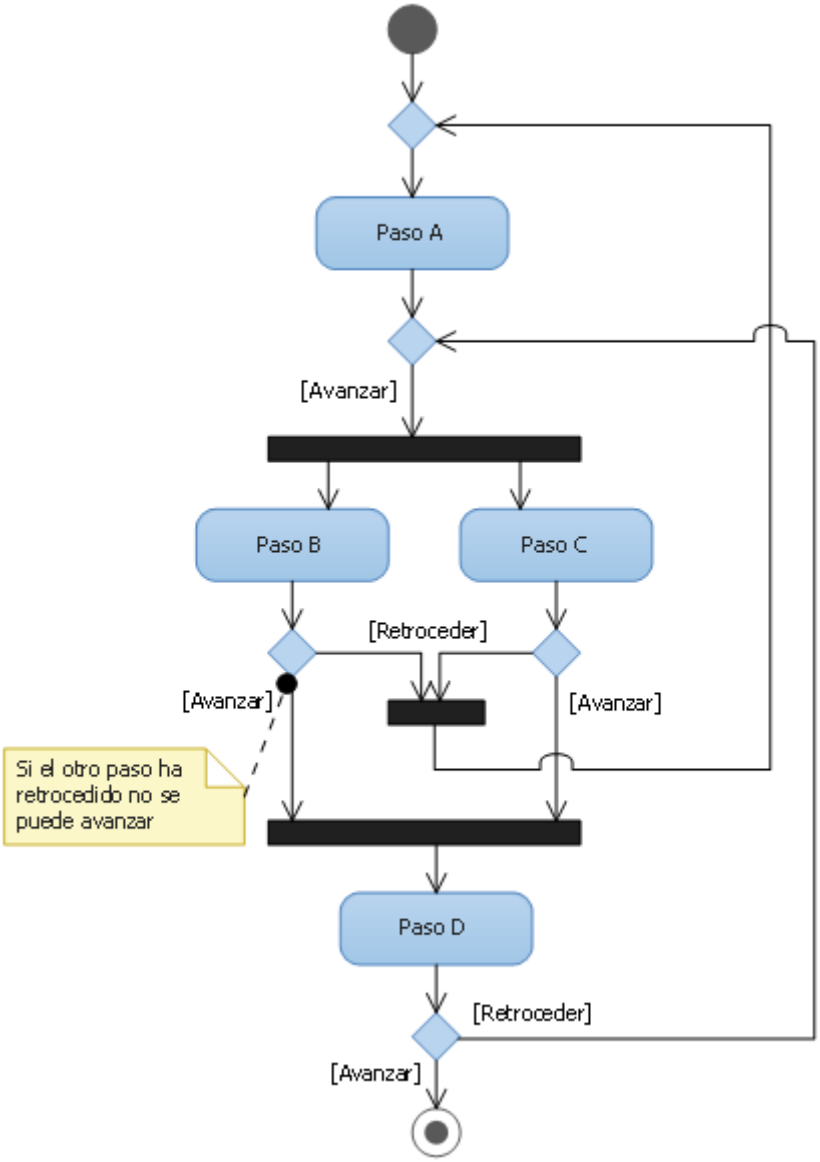


Figura 4: Ejemplo de workflow no lineal

3. WORKFLOW MANAGEMENT SYSTEM (WfMS)

3.1 Definición

La *Workflow Management Coalition* [1] define un *Workflow Management System* (WfMS) como:

"*un sistema que define, crea y gestiona la ejecución de workflows mediante el uso de uno o más* ***motores*** *software que se encargan de* ***interpretar*** *la definición de procesos,* ***interactuar*** *con los participantes y, cuando se requiera,* ***invocar*** *el uso de los sistemas informáticos implicados*" [3].

Implícitamente en la definición se presupone que existe un mecanismo o modelo para definir los *workflows* y que el WfMS lo interpreta.

3.2 Componentes

De todo lo anterior se deducen los componentes que son necesarios para desarrollar un WfMS:

- Modelo de *workflows*: los elementos y reglas que permiten representar, de una manera normalizada, los diferentes procesos de negocio de una empresa en un sistema informático.
- Motor de *workflows*: el núcleo del sistema, se encarga de interpretar el modelo de *workflows* y gestionar los estados de las distintas instancias de *workflows*.
- Interfaz de usuario: encargado de interactuar con los participantes. Permite la transmisión de información entre las instancias de *workflows* y los participantes y viceversa.
- Interfaz de conexión con otros sistemas: permite a los *workflows* realizar acciones utilizando otros sistemas, e.g. envío de notificaciones por mail, acceso a la lógica de negocio de la empresa, etc.

A continuación, se muestra un esquema en el que están representados los distintos componentes de los *workflows* y del WfMS y de cómo se relacionan entre ellos:

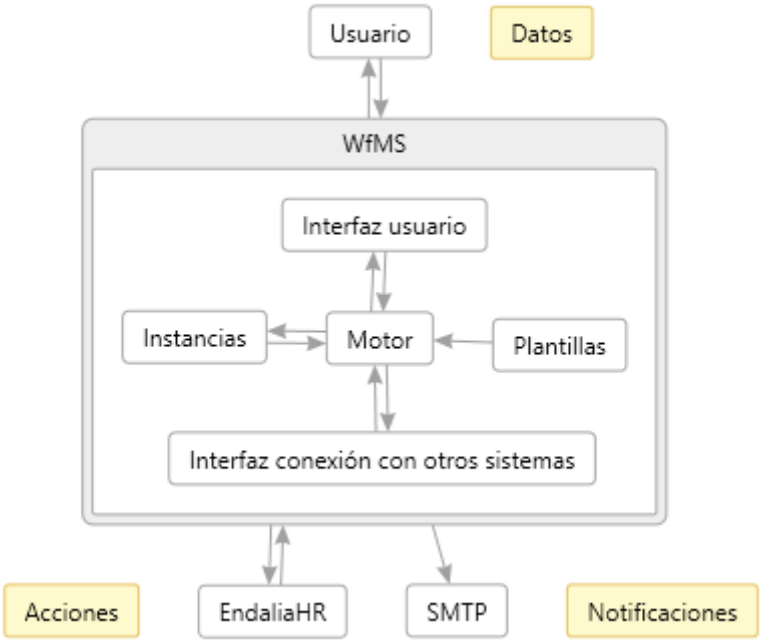


Figura 5: Esquema de los componentes de un WfMS

4. GLOSARIO

A continuación, se muestra un listado con los términos que se usarán a lo largo del proyecto para referirse a los distintos conceptos del sistema. Este listado se irá completando a lo largo de las primeras fases del proyecto:

Término	Definición
Workflow	Proceso que se está modelando en la aplicación.
Step	Cada una de las fases que componen un workflow.

Field	Cada uno de los campos que contienen información de un step.
Instancia	Una ejecución de un workflow.
Responsable	Los usuarios asignados como responsables de realizar un step.
Action	Las acciones que el sistema ejecuta automáticamente al avanzar o retroceder un paso de una instancia.
FinishCondition	Condición que debe cumplirse para permitir avanzar de paso.
Aprobar paso	Avanzar una instancia al siguiente step.
Rechazar paso	Retroceder una instancia al step anterior.
Cancelar instancia	Abortar la ejecución de una instancia.

5. BIBLIOGRAFÍA

[1] «WfMC,» [En línea]. Available: http://wfmc.org/.

[2] «Definición de Workflow,» [En línea]. Available: http://www.aiai.ed.ac.uk/project/wfmc/ARCHIVE/DOCS/glossary/glossary.html#RTFToC9.

[3] «Definición de WfMS,» [En línea]. Available: http://www.aiai.ed.ac.uk/project/wfmc/ARCHIVE/DOCS/glossary/glossary.html#RTFToC13.

endalia®
HR Consulting & Software

II. ESTUDIO DE MERCADO

WORKFLOW MANAGEMENT SYSTEM

DAVID ROY

TABLA DE CONTENIDOS

Tabla de contenidos 2
1. Introducción 3
1.1 Objetivo 3
1.2 Alcance 3
1.3 Organización del documento 3
2. lenguajes de modelado 3
2.1 BPMN 3
2.2 XPDL 5
2.3 WS-BPEL 5
2.4 Conclusiones 6
3. Mercado actual 6
3.1 jBPM 7
3.1.1 Diseñador 7
3.1.2 Interfaz de usuario 8
3.1.3 Interconexiones 9
3.2 Bonita BPM 10
3.2.1 Diseñador 10
3.2.2 Interfaz de usuario 11
3.2.3 Interconexiones 12
3.3 ProcessMaker 13
3.3.1 Diseñador 13
3.3.2 Interfaz usuario 14
3.3.3 Interconexión 15
3.4 WorkflowGen 16
3.4.1 Diseñador 16
3.4.2 Interfaz de usuario 18
3.4.3 Interconexiones 20
3.5 KissFlow 21
3.5.1 Diseñador 21
3.5.2 Interfaz de usuario 23
3.5.3 Interconexiones 24
3.6 Conclusiones 25
4. Bibliografía 25

1. INTRODUCCIÓN

1.1 Objetivo

El objetivo de este estudio es analizar tanto los estándares existentes de notación y modelado de workflows, como las herramientas informáticas del mercado cuya finalidad es la gestión de *workflows*. La intención del estudio es detectar características y funcionalidades que deban ser aplicadas en el sistema que se va a desarrollar.

1.2 Alcance

Se analizarán algunas de las aplicaciones software más destacadas del mercado y también los formatos y estándares más importantes.

1.3 Organización del documento

El presente documento contiene las siguientes secciones:

1. Introducción: Descripción del objetivo y alcance del documento.
2. Lenguaje de modelado: repaso de los diferentes estándares existentes para representar *workflows*.
3. Mercado actual: estudio de diferentes herramientas actuales.
4. Bibliografía: Referencias a libros y webs utilizadas en el documento.

2. LENGUAJES DE MODELADO

Un concepto fundamental en el ámbito de los *workflows* es el modelado o definición de los mismos, esto es, la representación de un proceso del mundo real en un entorno informatizado. En la actualidad existen varios formatos para representar flujos de trabajo, tanto notaciones gráficas como lenguajes. En esta sección se van a analizar los principales, tanto sus características como su aceptación entre las herramientas de gestión de *workflows* existentes en el mercado.

2.1 BPMN

Business Process Model and Notation (BPMN) [1] es una notación gráfica estandarizada que permite el modelado de workflows. Inicialmente fue desarrollada por la organización *Process Management Initiative* (BPMI) y en la actualidad es mantenida por el *Object Management Group* (OMG) tras la fusión de ambas organizaciones en 2005.

Debido a la adopción cada vez mayor de la notación BPMN entre los sistemas de gestión de *workflows*, se puede considerar el estándar *de facto*.

Algunos de los componentes principales de los diagramas BPMN son:

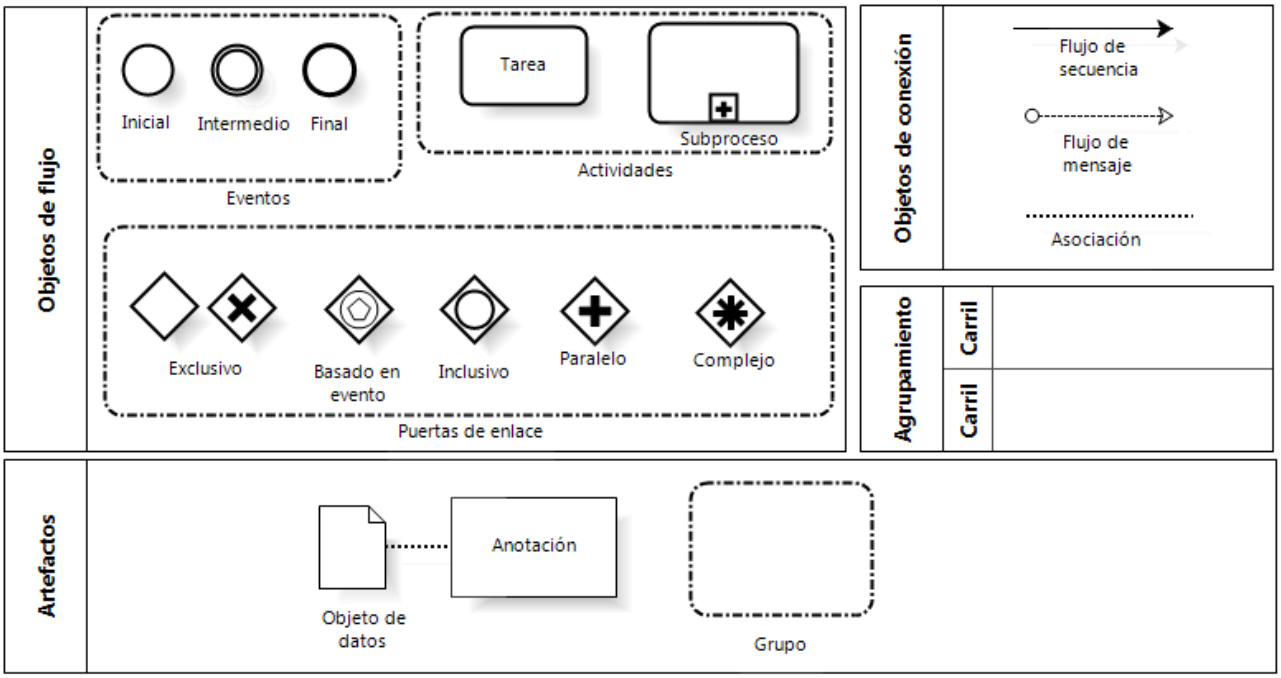


Figura 1: Componentes de diagramas BPMN

Se va a detallar brevemente qué significa y para qué se usa cada uno de ellos:

- Objetos de flujo:
  - Eventos: describen “algo que sucede”.
  - Actividades: describen “algo que se hace”.
    - Tarea: la unidad mínima.
    - Subproceso: una agrupación de tareas.
  - Compuertas: representados por un rombo. Describen una condición y permiten bifurcar o combinar rutas.
- Objetos de conexión: permiten conectar los objetos
  - Secuencia: describe el orden de ejecución.
  - Mensaje: describe un mensaje enviado de objetos de un carril a otro distinto.
  - Asociación: se utilizan para relacionar artefactos y objetos.
- Agrupamientos: sirven para organizar los elementos.
  - Carril: unidad mínima de agrupamiento.
  - Piscina: puede contener carriles.
- Artefactos: se conectan a los objetos con conexiones de asociación y aportan más información.
  - Objeto de datos: indica el dato requerido o producido por una actividad.
  - Grupos: sirven para agrupar visualmente diferentes actividades.
  - Anotación: sirven para añadir comentarios o descripciones.

A continuación, se muestra un ejemplo de diagrama BPMN:

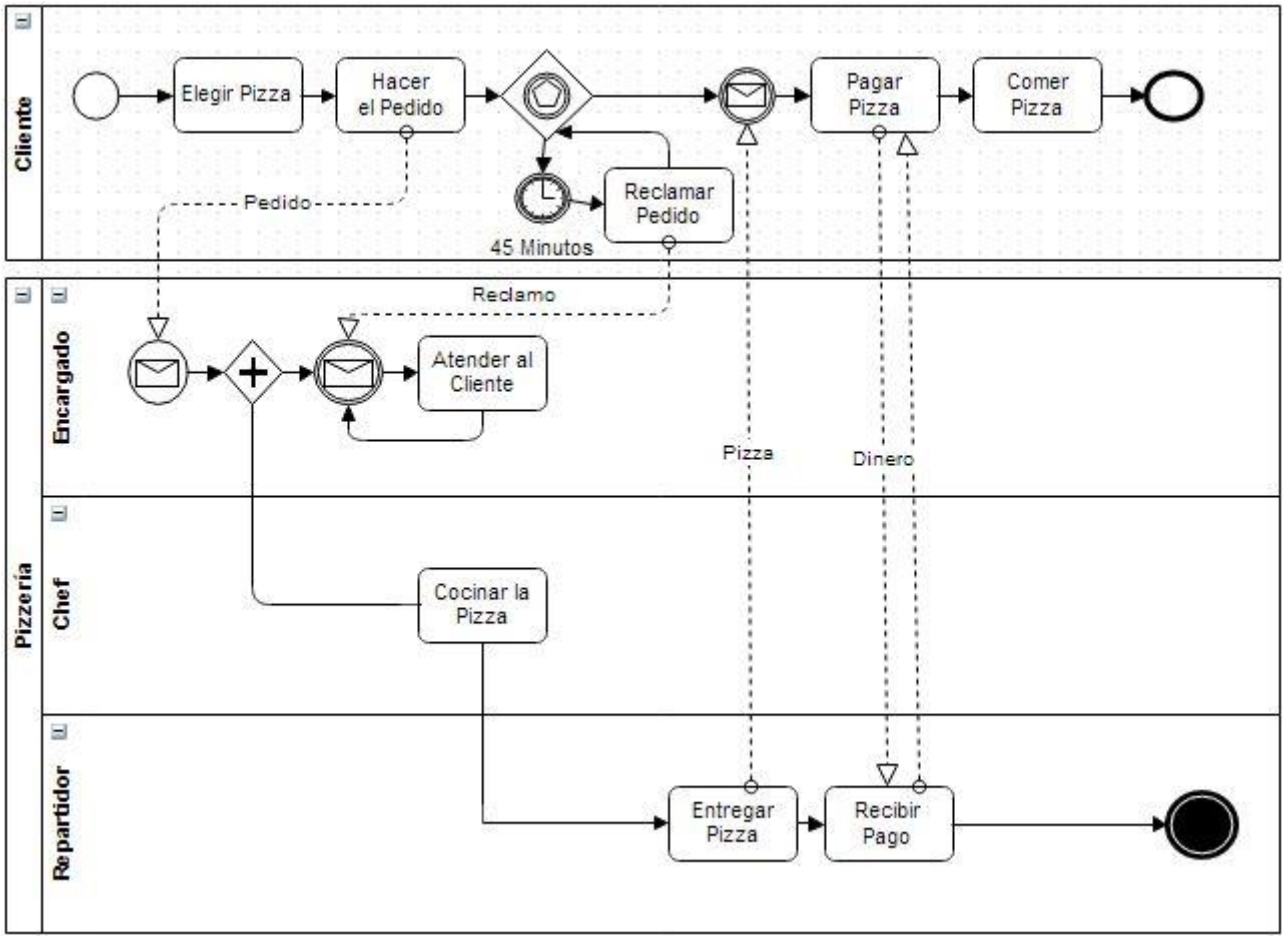


Figura 2: Ejemplo diagrama BPMN

2.2 XPDL

XML Process Definition Language (XPDL) [2] es un leguaje de definición de *workflows* creado en el año 2001 por la *Workflow Management Coalition* (WfMC) [3].

La característica más destacada del formato XPDL es que además de guardar la información que define el *workflow* también guarda la información para representarlo gráficamente. Uno de sus objetivos es precisamente, ser el formato estándar de serialización de los diagramas BPMN.

2.3 WS-BPEL

Web Services Business Process Execution Language (WS-BPEL) [4] es un lenguaje de definición de *workflows* estandarizado por la *Organization for the Advancement of Structured Information Standards* (OASIS), un consorcio internacional sin ánimo de lucro que se orienta al desarrollo, la convergencia y la adopción de los estándares de comercio electrónico y servicios web.

Se diferencia de XPDL en que WS-BPEL no almacena información sobre la representación gráfica del *workflow*. Por otro lado, WS-BPEL es un lenguaje mucho más potente que XPDL para representar la parte funcional del *workflow*. En WS-BPEL un *workflow* se representa mediante servicios web, que son las acciones

que se ejecutan en cada paso, y mediante variables que determinan cuándo y cómo se invocan estos servicios.

2.4 Conclusiones

Los 3 estándares estudiados son complementarios entre sí y pueden utilizarse conjuntamente. Así pues, un *workflow* podría representarse gráficamente mediante un diagrama BPMN, guardarse en un fichero XPDL y las reglas de ejecución del *workflow* podrían guardarse en lenguaje BPEL para ser ejecutadas por un motor de *workflows* compatible.

Implementar el modelo completo BPMN queda fuera del alcance del proyecto, por lo que se tomará un subconjunto reducido de elementos que se implementarán en el nuevo sistema:

- Eventos: inicio de *workflow*, fin de *workflow*, envío de email o ejecución. Se representan con un círculo.
- Actividades: los pasos o fases del *workflow*. Se representan con un rectángulo.
- Flujo de secuencia: indica el orden de ejecución de los pasos. Se representan con una flecha.
- Compuerta paralela: permite bifurcar o juntar el flujo de secuencia. Se representa con un rombo.
- Artefactos: información que se recoge en un paso. Se representan mediante un rectángulo y una flecha discontinua que lo une al paso.

En futuras extensiones del sistema, podría abordarse la implementación completa de BPMN.

Con respecto al formato XPDL, en los requisitos no se considera la posibilidad de exportar los *workflows* a otro sistema ni importarlos, por lo tanto, el formato XPDL no se va a aplicar en este proyecto.

Por otro lado, el motor de *workflows* que se va a desarrollar no va a soportar el lenguaje de orquestación WS-BPEL porque debido a su complejidad queda fuera del alcance del proyecto. El motor de *workflows* que se va a implementar estará integrado en el sistema EndaliaHR como un módulo de código, por tanto, las tareas podrán ejecutarse en EndaliaHR a través de llamadas a procedimientos y funciones, sin requerir el uso de servicios web.

3. MERCADO ACTUAL

Para evaluar las herramientas actuales del mercado, se va a ejecutar en cada una de ellas un mismo workflow de ejemplo: “Solicitud de día libre” que constará de 2 pasos: solicitud y aprobación. En el primer paso se recogen 2 campos de información, nombre del empleado y fecha, y en el segundo paso se aprueban. Al finalizar se envía un email al usuario para notificarle si el workflow ha sido aprobado o no.

De cada herramienta se valorará su funcionalidad, su interfaz de usuario y sus posibilidades de conexión con otros sistemas. Además, se destacarán aquellas funcionalidades que parezcan útiles para el sistema. Al final de la sección se recopilarán las conclusiones del estudio.

3.1 jBPM

jBPM [5] es un proyecto de código abierto desarrollado por Red Hat y la comunidad jBoss. Está escrito en java y se ejecuta bajo entorno web.

3.1.1 Diseñador

Entre sus funcionalidades dispone de un diseñador gráfico de workflows que es compatible con la notación BPMN. El diseñador es muy fácil de manejar y muy cómodo: desde cada nodo puedes crear el siguiente (no hay que arrastrarlo desde el *toolbox*).

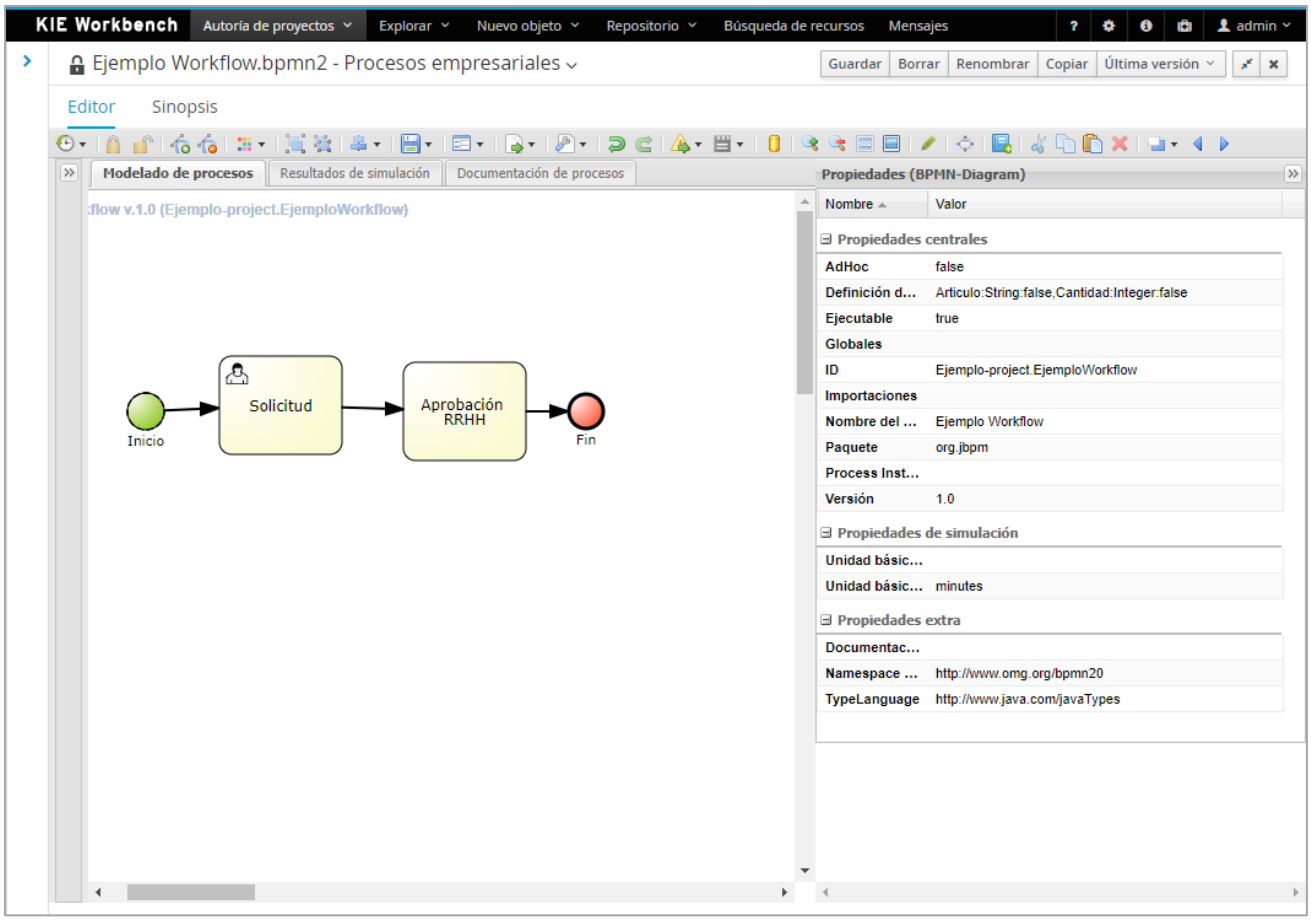


Figura 3: Captura de pantalla de jBPM

En cada paso se asignan unos permisos de usuario para que puedan ejecutarlo o se pueden asignar directamente usuarios específicos. La aplicación dispone de un menú de administración para gestionar usuarios y permisos.

En cada actividad se puede asociar un formulario para mostrar al usuario:

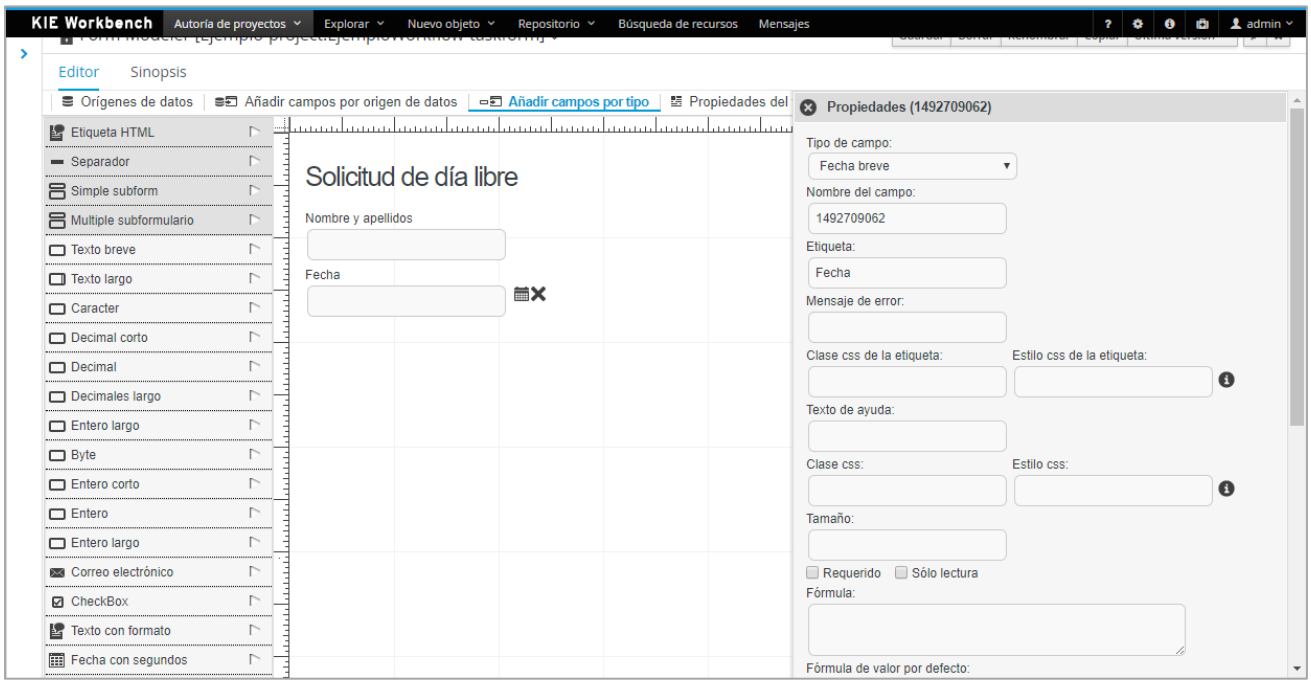


Figura 4: Diseñador de formularios de jBPM

Los formularios se componen de campos, cada uno con su etiqueta y su tipo. Se destaca la gran variedad de tipos de campos que tiene, así como las opciones de marcar un campo como “Requerido” o “Sólo lectura”.

3.1.2 Interfaz de usuario

El interfaz tiene un diseño actual, está disponible en varios idiomas y su funcionamiento es muy fluido. Aunque dispone de una vista “Básica” que es más simplificada que la vista “Avanzada”, sigue estando muy recargada de opciones y menús por lo que no resulta fácil de manejar y requiere tiempo para aprender su funcionamiento.

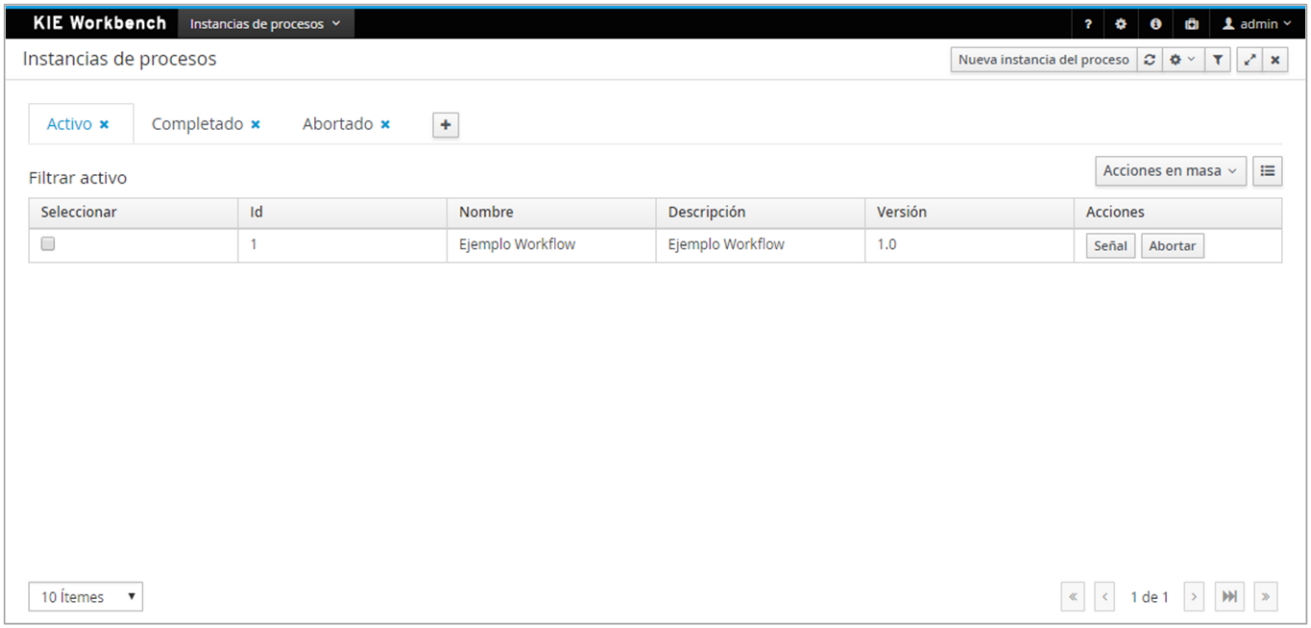


Figura 5: Pantalla de Instancias de procesos de jBPM

Entre las pantallas de usuario, se destaca la de “Instancias de procesos” que muestra un listado con las instancias y permite filtrarlas por las que siguen activas, las que han finalizado o las que han sido canceladas. Desde esa misma pantalla se puede abrir cada una de las instancias y acceder a su formulario o también se puede iniciar una nueva instancia.

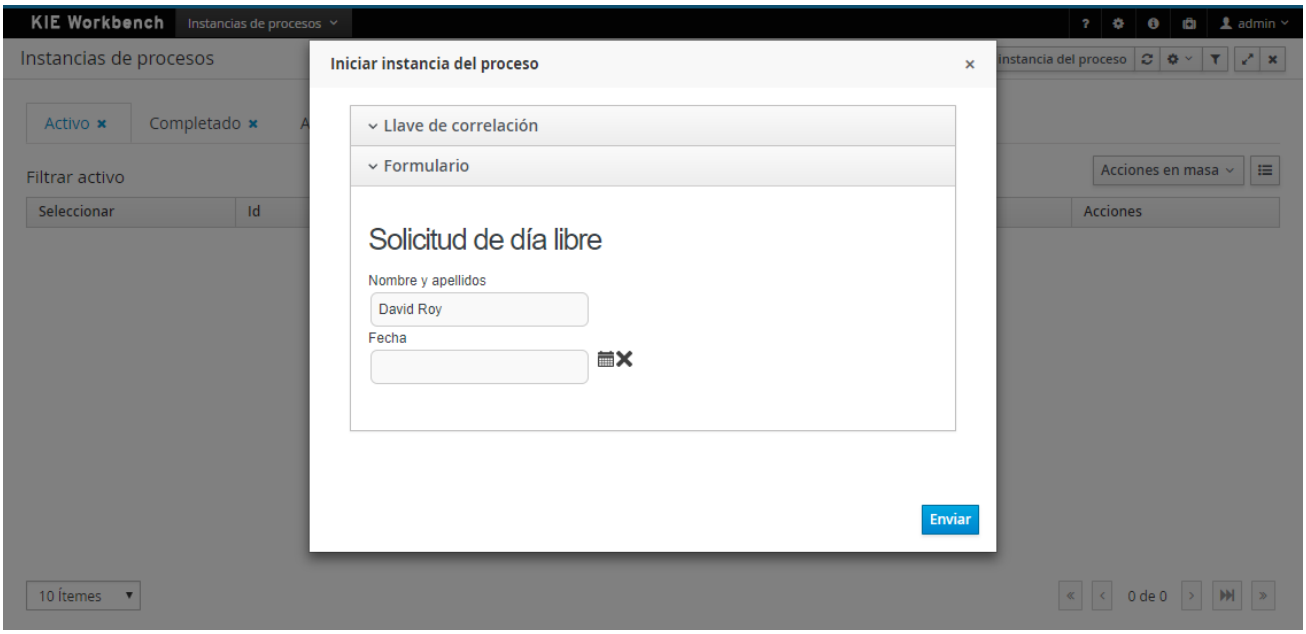


Figura 6: Iniciar una nueva instancia desde jBPM

3.1.3 Interconexiones

La aplicación permite que cada paso del workflow pueda realizar interconexiones con otros sistemas. Se pueden realizar envíos de email, conectar con un *webservice* o realizar una llamada a una Api REST.

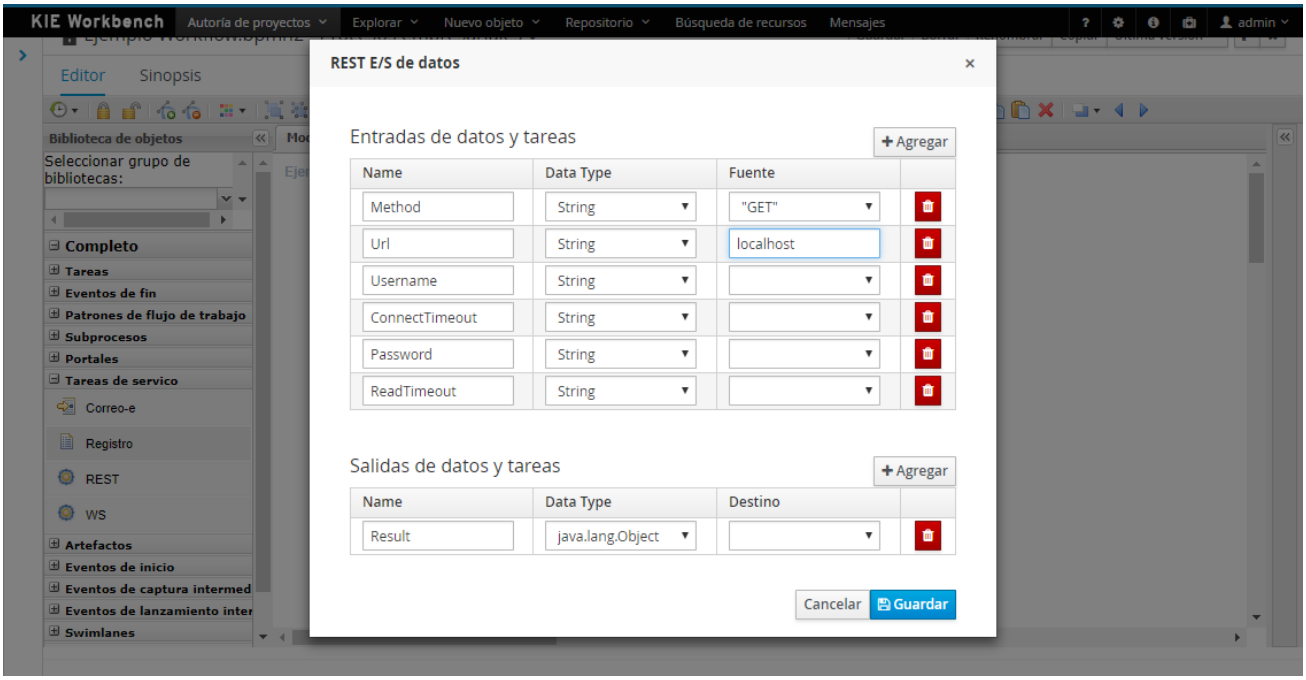


Figura 7: Configuración interconexiones de un workflow jBPM con otros sistemas

3.2 Bonita BPM

Bonita BPM [6] es un proyecto de código abierto creado en 2001. Inicialmente fue desarrollado por el *Institut national de recherche en informatique et en automatique* (INRIA) y desde 2009 es mantenido por la compañía Bonitasoft.

Está escrito en java y dispone de aplicación de escritorio y aplicación web. La aplicación de escritorio se utiliza para la administración del sistema y la web es el portal a través del que acceden los usuarios.

3.2.1 Diseñador

Bonita BPM dispone de un diseñador gráfico de workflows, que se ejecuta desde la aplicación de escritorio:

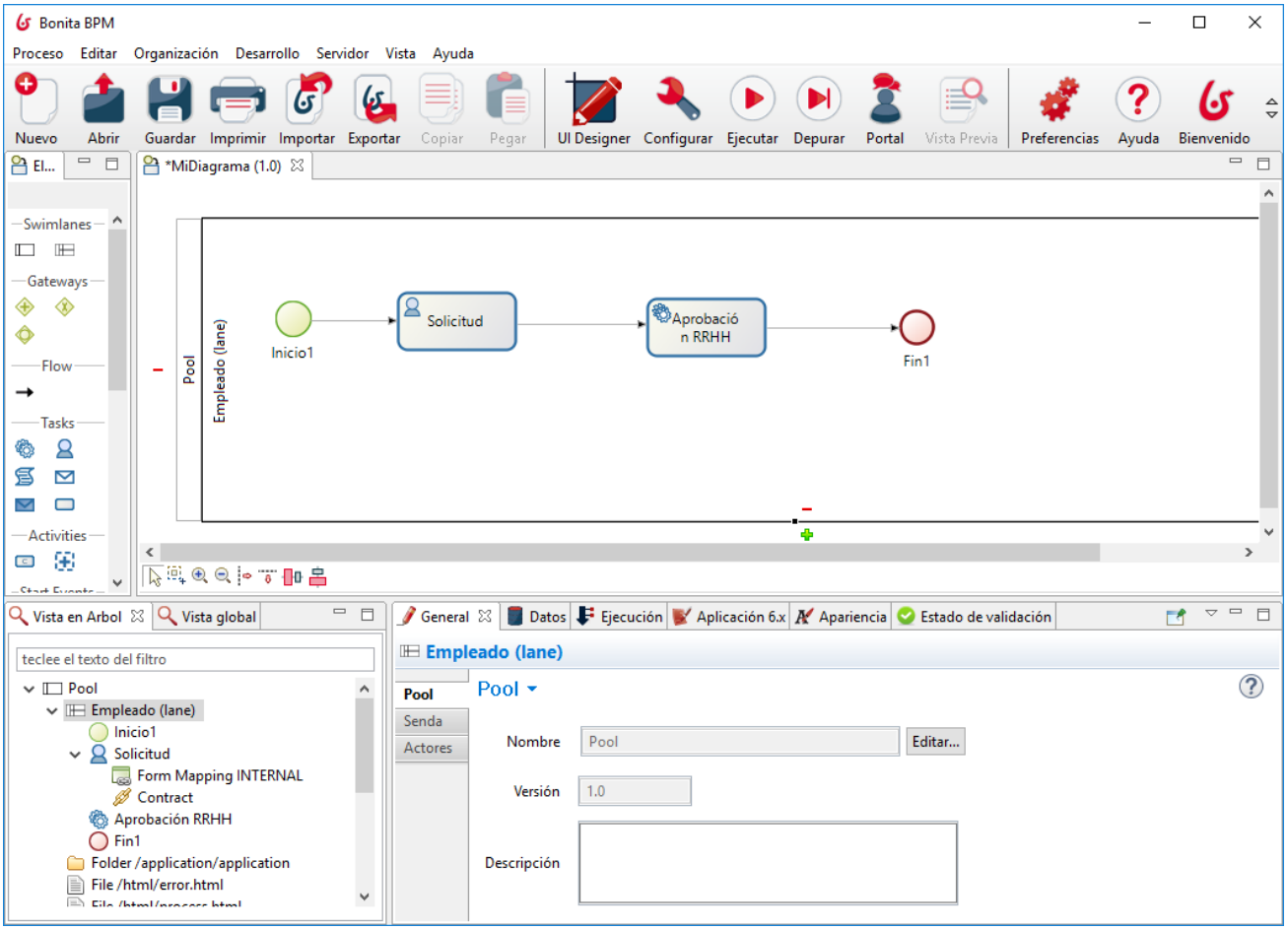


Figura 8: Diseñador gráfico de workflows de BonitaBPM

El diseñador de workflows es compatible con el estándar BPMN y al igual que jBPM es muy fácil y cómodo de manejar. También dispone de un diseñador de formularios, para asociarlos al workflow. En este caso, el diseñador de formularios que se ha probado es el que recomienda la aplicación y se ejecuta a través de la web, aunque dispone de otra versión que se ejecuta en el escritorio.

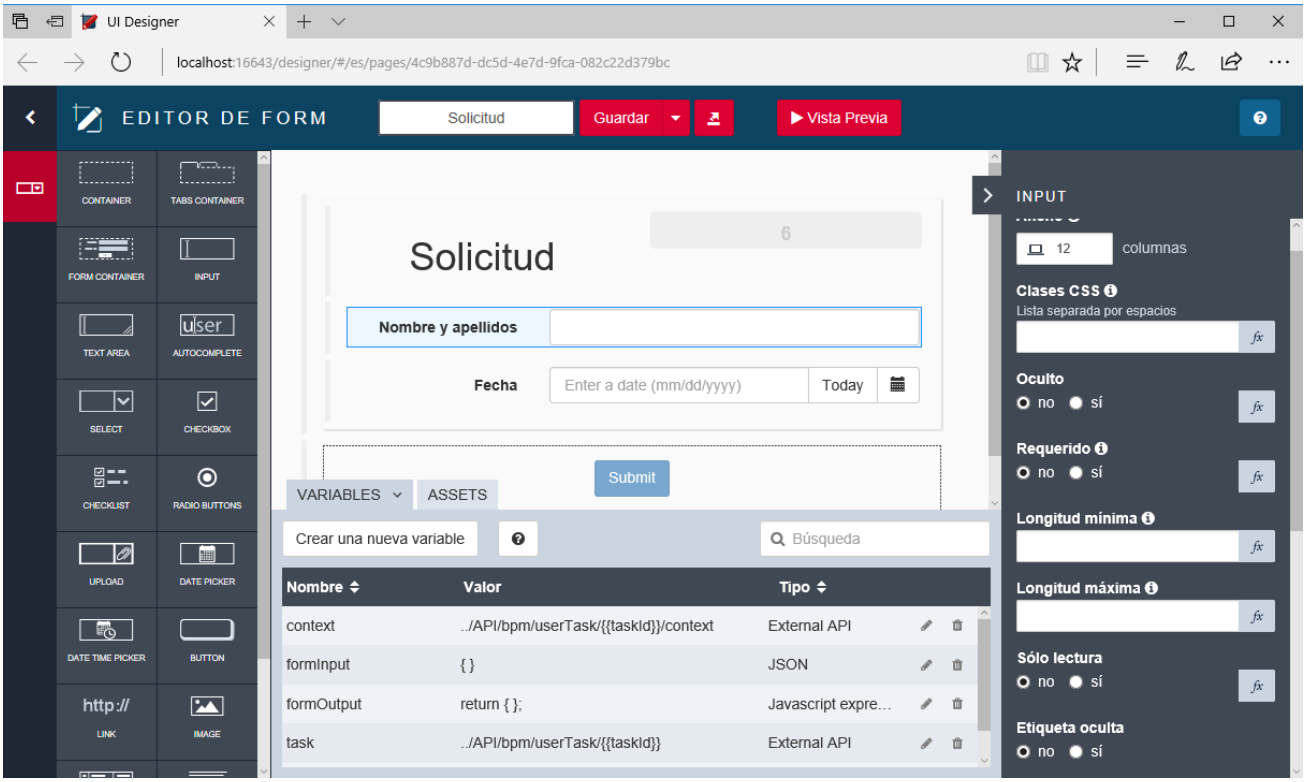


Figura 9: Diseñador de formularios de BonitaBPM

El diseñador de formularios tiene la misma funcionalidad que el de la herramienta que se ha estudiado anteriormente. Se destaca la opción de “ocultar” campos según una condición que podemos definir.

3.2.2 Interfaz de usuario

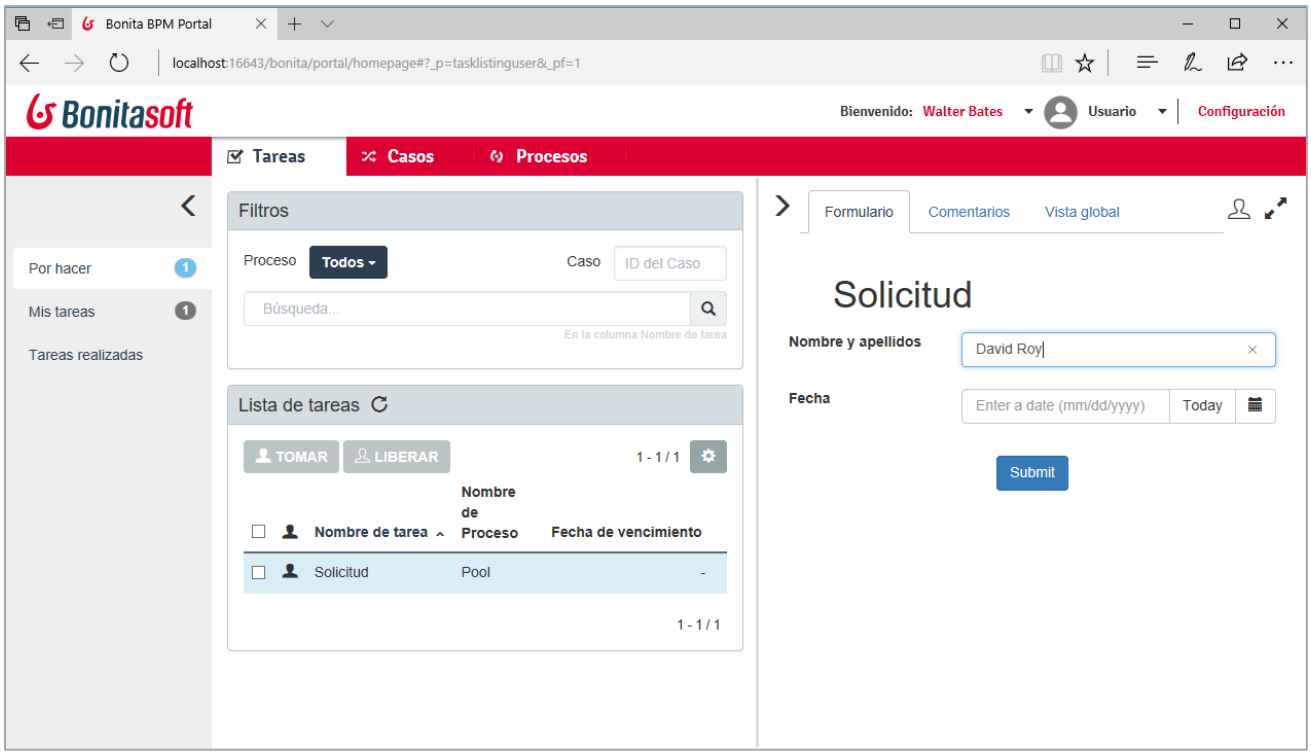


Figura 10: Pantalla principal de usuario de Bonita BPM

La interfaz de usuario tiene un diseño actual, fluido y destaca por su sencillez, lo que la hace muy fácil de manejar desde el primer momento.

La pantalla principal muestra una lista de procesos que puedes filtrar por pendientes y finalizados. Desde esa misma pantalla puedes acceder a cada uno de los procesos y sus formularios sin perder de vista el listado principal.

3.2.3 Interconexiones

Se ha comprobado que Bonita BPM dispone de envío de emails a través de un servidor SMTP para notificar a los usuarios cuando un workflow avanza de fase.

Para interconectar los workflows con otros sistemas, Bonita BPM puede ejecutar consultas SQL sobre una base de datos externa que podemos configurar.

3.3 ProcessMaker

ProcessMaker [7] es una aplicación comercial de código abierto desarrollada por la empresa ProcessMaker Inc. que se ejecuta en entorno web.

3.3.1 Diseñador

ProcessMaker tiene un diseñador gráfico de workflows compatible con el estándar BPMN. Es muy intuitivo, tiene vídeos y animaciones de ayuda y destaca el auto-corrector que te indica si el diagrama que estás haciendo no cumple las normas BPMN.

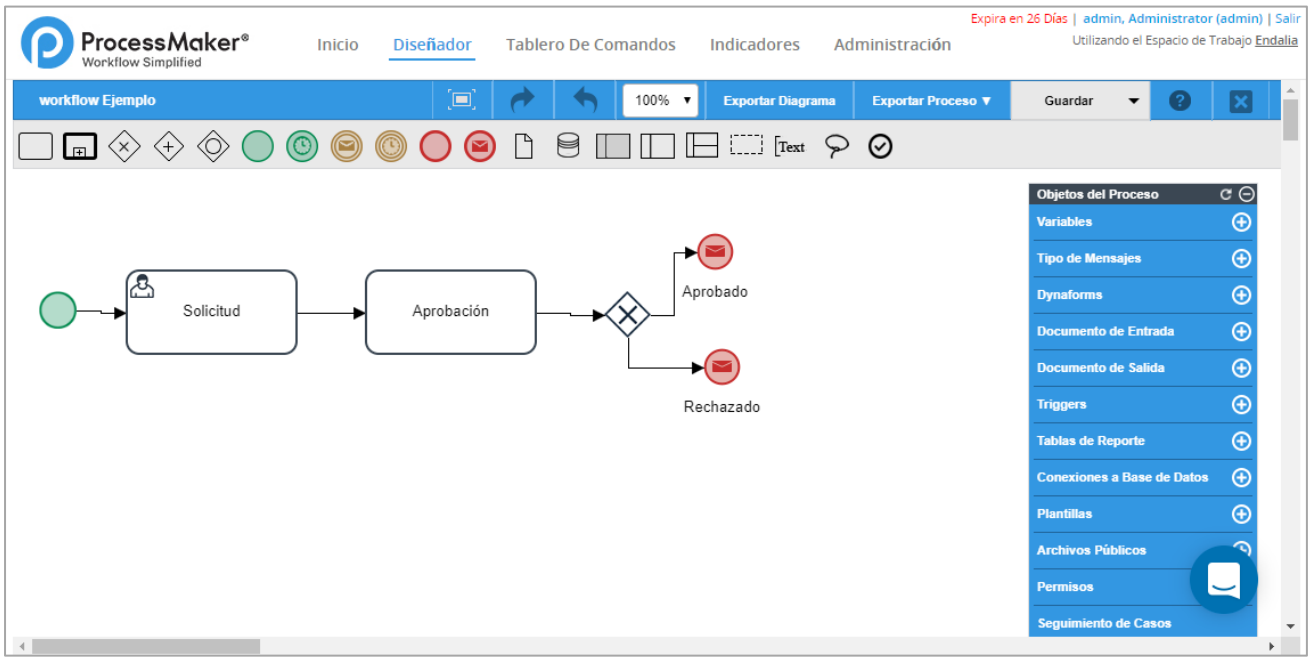


Figura 11: Diseñador de workflows de ProcessMaker

De manera similar a las herramientas estudiadas anteriormente, ProcessMaker permite añadir formularios al workflow y dispone de un diseñador de formularios para crearlos:

Figura 12: Diseñador de formularios de ProcessMaker

3.3.2 Interfaz usuario

La pantalla principal está organizada como si fuera un gestor de correo. Los workflows se muestran en una tabla principal donde puedes ordenarlos y filtrarlos por diferentes criterios.

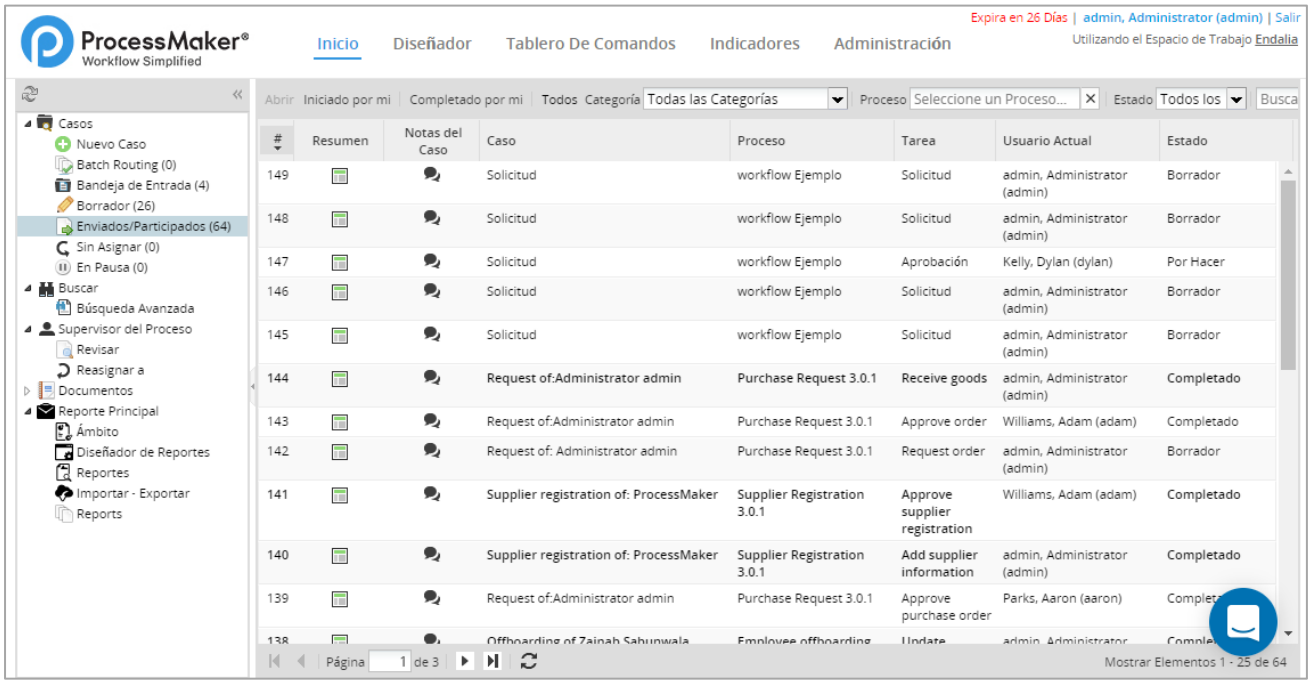


Figura 13: Pantalla principal de ProcessMaker

Desde la pantalla principal puedes iniciar nuevos workflows:

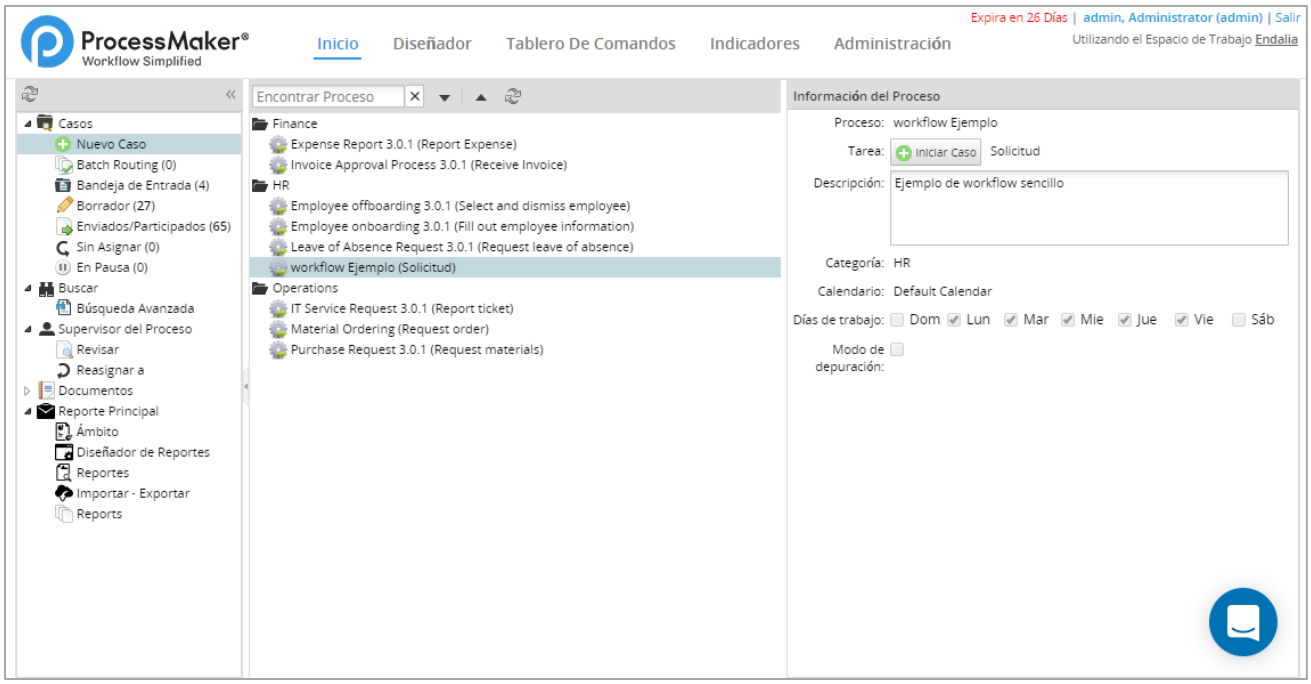


Figura 14: Iniciar nuevo workflow desde ProcessMaker

Se destaca que los workflows están agrupados en categorías.

Una vez iniciado el workflow, se muestra su formulario asociado para que el usuario complete la información:

Figura 15: Formulario de ProcessMaker

Se destaca que el usuario también puede ver el "Mapa del proceso" en el que se le muestra el diagrama del workflow resaltando los pasos que están en curso, finalizados, etc.

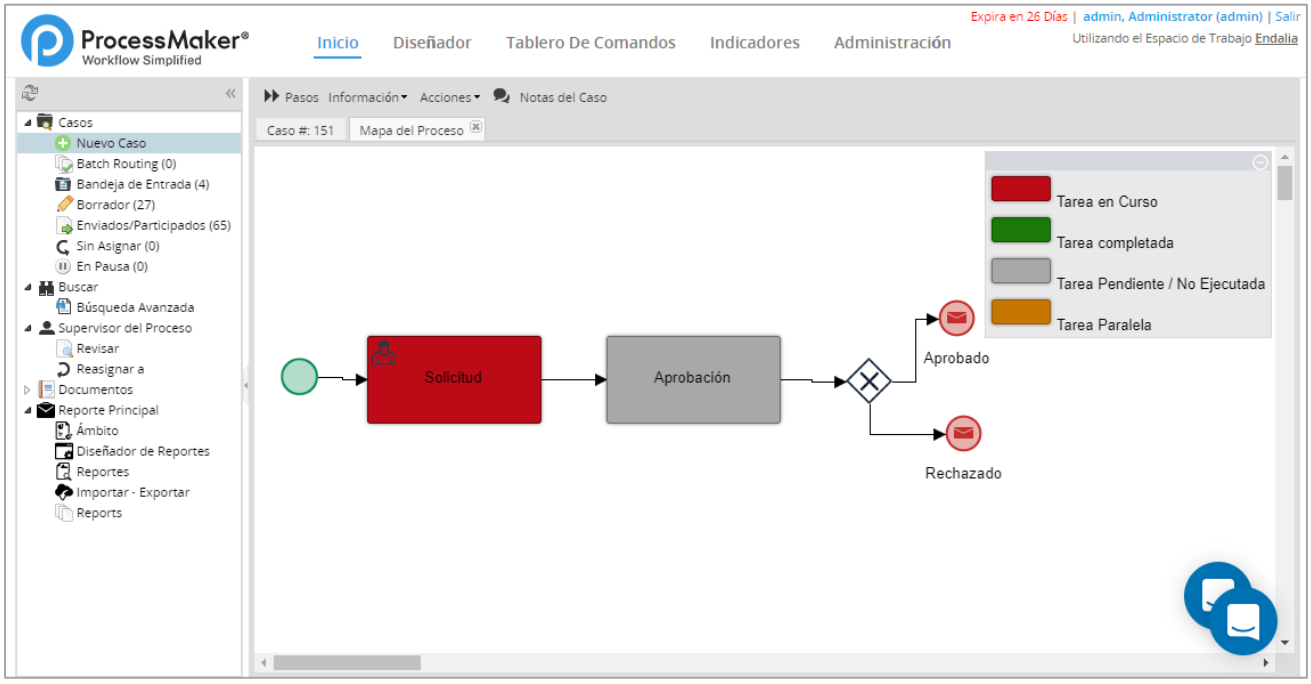


Figura 16: Mapa del proceso de ProcessMaker

3.3.3 Interconexión

ProcessMaker puede conectarse a bases de datos externas, aunque sólo admite *MySql*, para ejecutar consultas desde un workflow.

También puede conectarse con un servidor SMTP para que los workflows envíen emails de notificaciones.

3.4 WorkflowGen

WorkflowGen [8] es una aplicación comercial de gestión de workflows creada por la compañía Advantys. Está desarrollada sobre la plataforma .NET y está basada en entorno web.

3.4.1 Diseñador

WorkflowGen dispone de un diseñador gráfico para definir los workflows. Comparado con los productos anteriores, éste diseñador es más básico y cuenta con menos opciones, además no implementa BPMN.

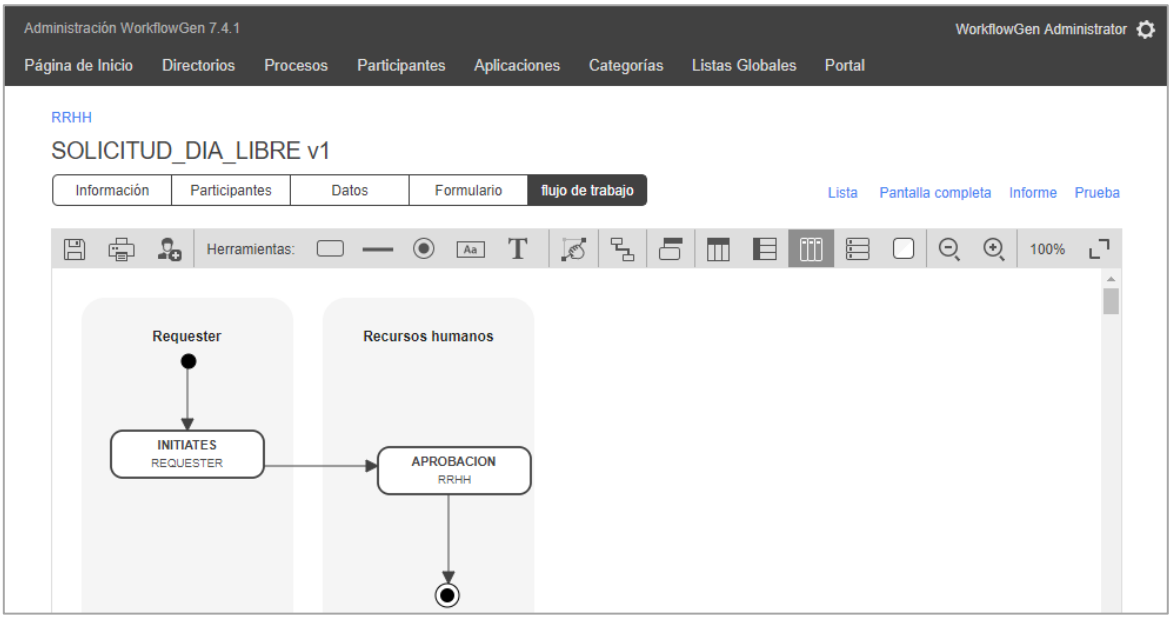


Figura 17: Diseñador gráfico de workflows de WorkflowGen

El diseñador de formularios también es más básico que los de las otras aplicaciones estudiadas.

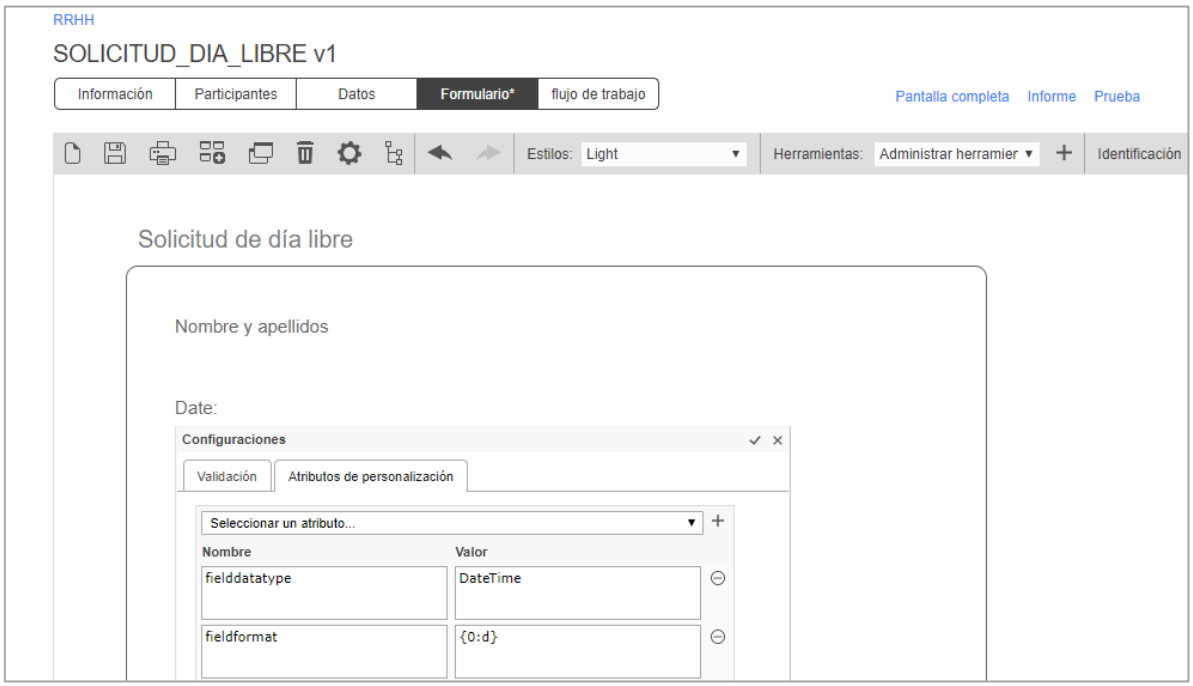


Figura 18: Diseñador de formularios de WorkflowGen

A pesar de ser más básico que otras herramientas, WorkflowGen incorpora algunas características interesantes como son:

- Gestión de delegaciones: permite que un usuario delegue sus workflows a otro usuario durante un determinado periodo temporal. Un ejemplo de uso de esta funcionalidad sería cuando un responsable de aprobar determinados workflows delega esta tarea durante sus vacaciones para que no se retrasen las aprobaciones.
- Definir los formularios en varios idiomas. Esta funcionalidad puede ser muy útil cuando la organización cuenta con empleados de diferentes países o que hablan diferentes lenguas (catalán, gallego, etc.).
- Compatibilidad con el formato XPDL.

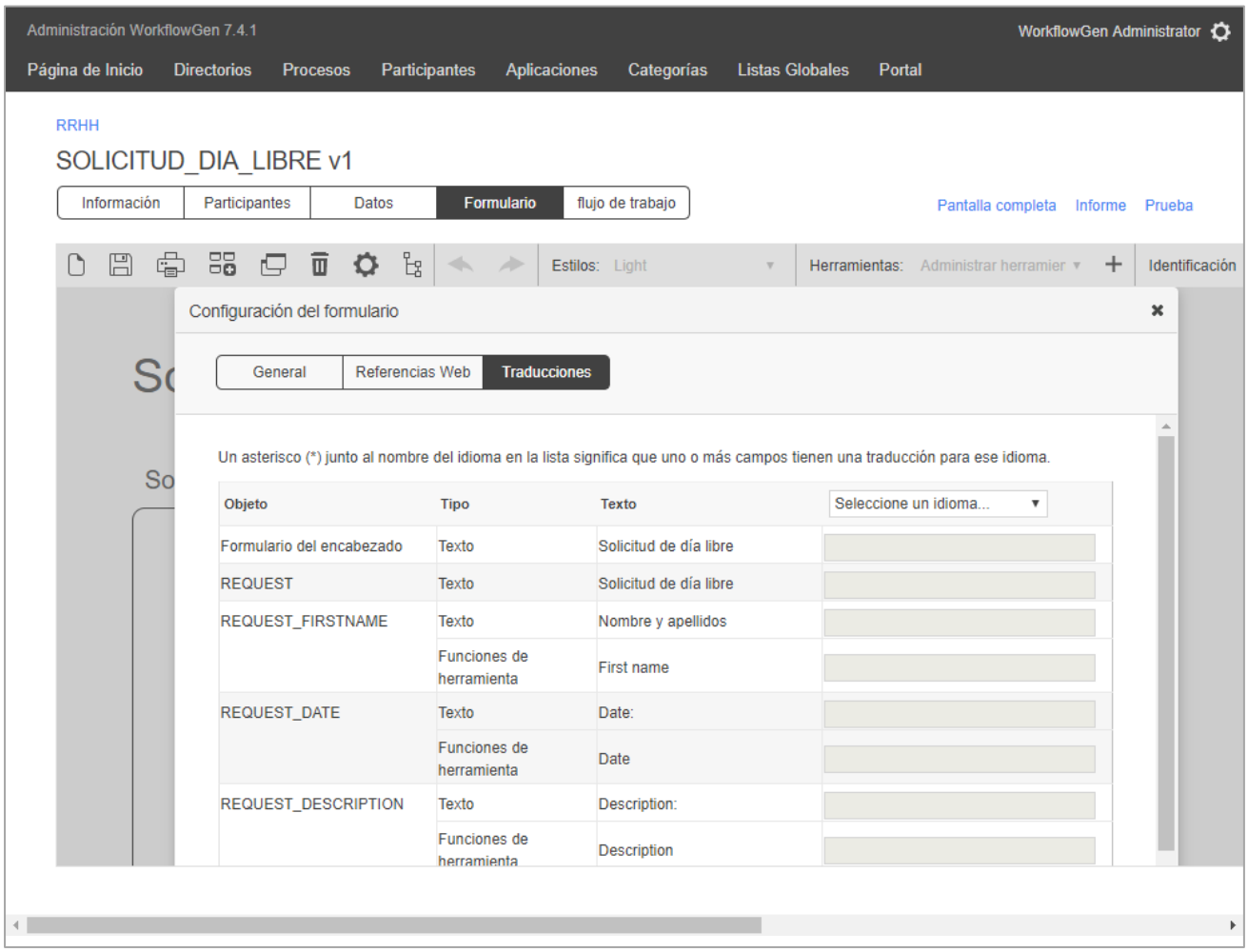


Figura 19: Formularios multi-idioma en WorkflowGen

3.4.2 Interfaz de usuario

WorkflowGen está disponible en varios idiomas. Como se ha indicado en el punto anterior, su interfaz de usuario es más básica que la de otras herramientas. Esto tiene una parte positiva porque la pantalla principal está menos recargada y resulta ser realmente sencilla de entender y manejar.

En la pantalla principal se muestran varias tablas con los workflows que el usuario ha iniciado y con los workflows en los que el usuario participa o tiene tareas pendientes.

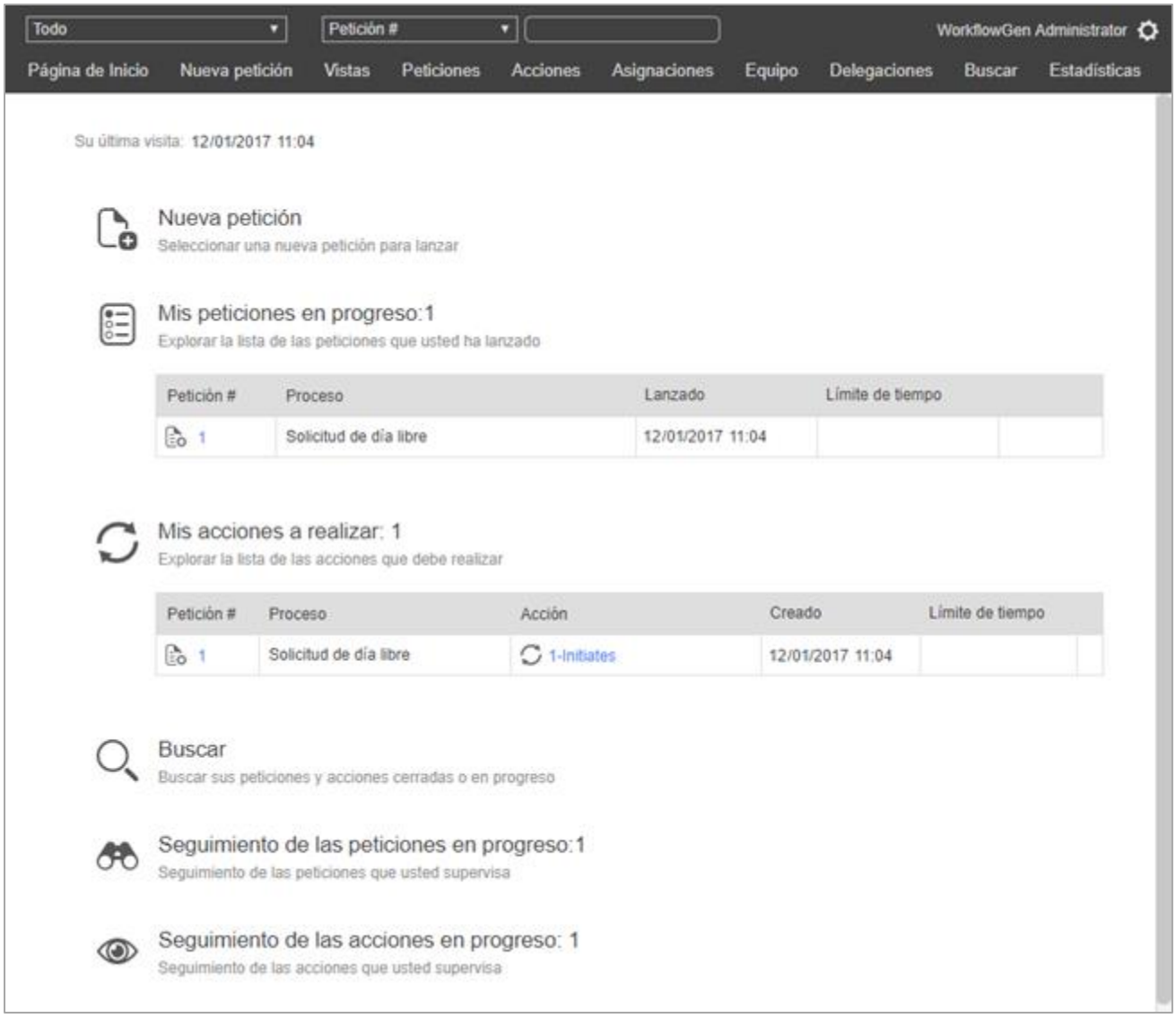


Figura 20: Pantalla principal de WorkflowGen

Desde la pantalla principal se puede iniciar un nuevo workflow o abrir uno existente:

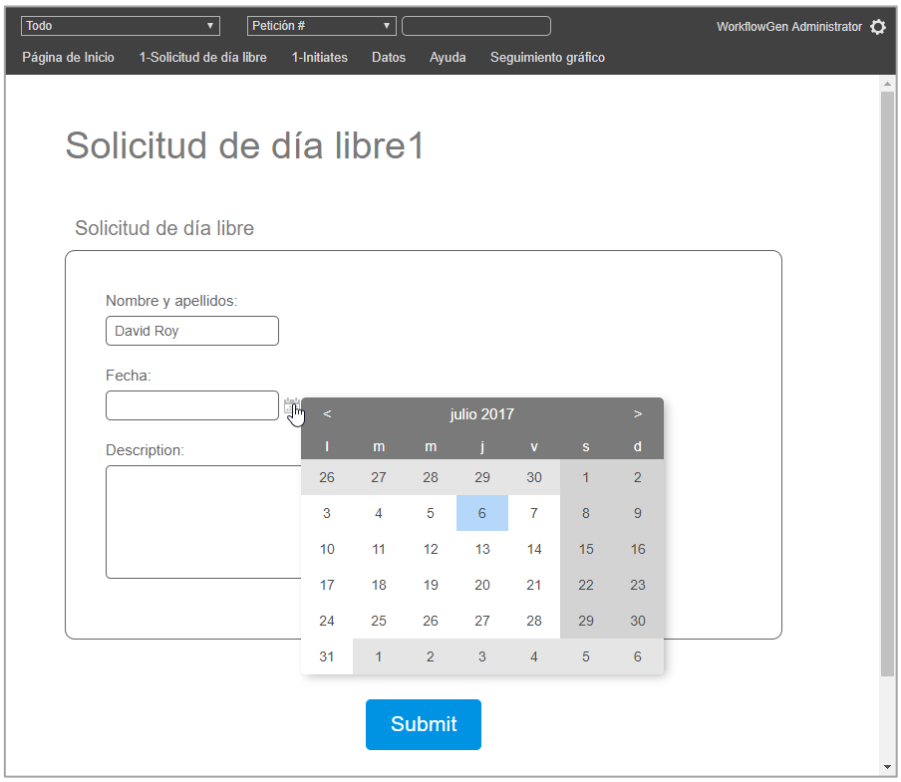


Figura 21: Formulario de workflow en WorkflowGen

El usuario dispone de la opción “Seguimiento gráfico” que le muestra el diagrama del workflow, resaltando el paso que está en curso:

Figura 22: Diagrama de workflow en WorkflowGen

3.4.3 Interconexiones

WorkflowGen permite conectarse con otras aplicaciones mediante servicios web y recibir mensajes desde otros sistemas mediante *webhooks*[1]:

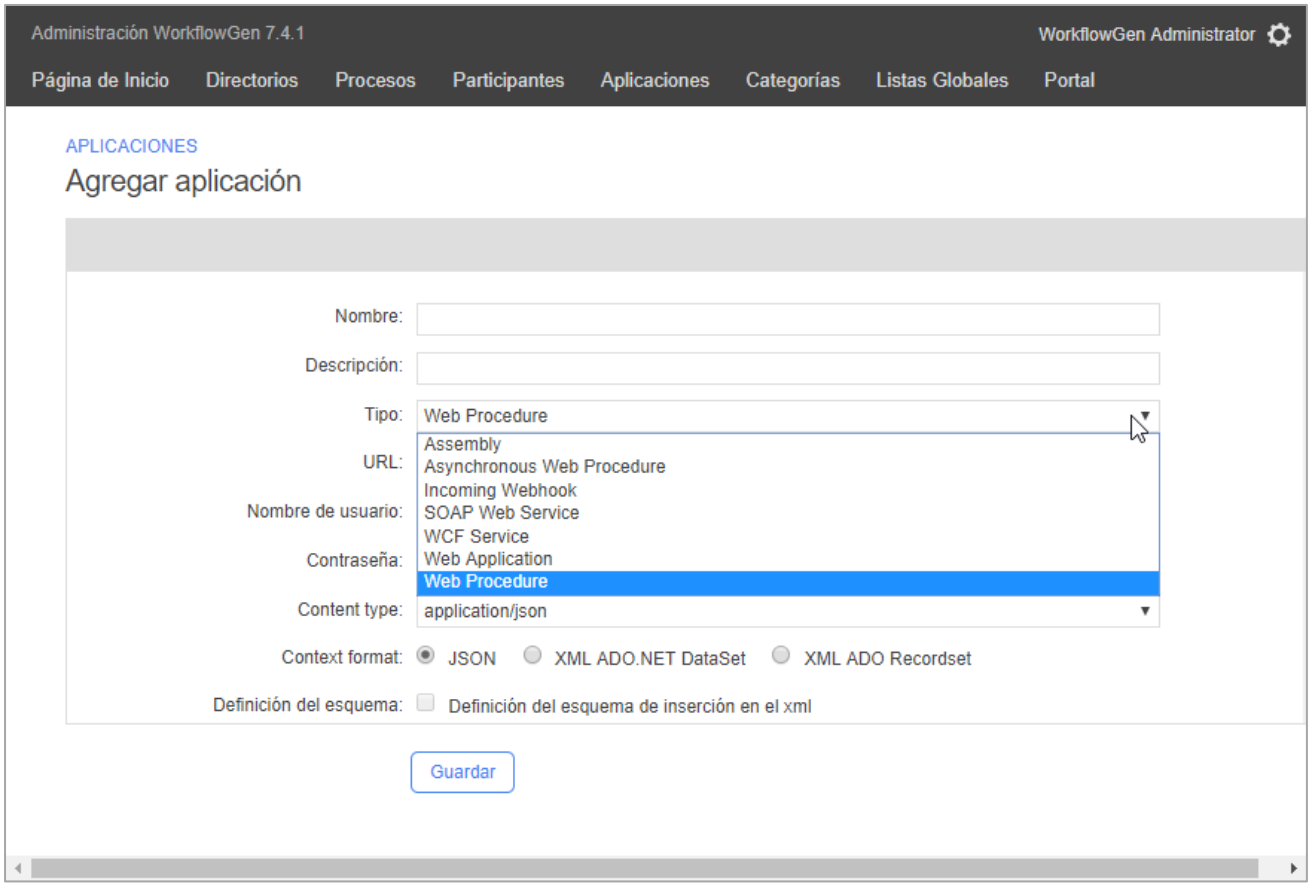


Figura 23: Conexión de WorkflowGen con aplicaciones de terceros

[1] *Webhook*: una llamada HTTP a una determinada url que está preparada para desencadenar una determinada acción al recibir una conexión.

3.5 KissFlow

KissFlow [9] es una aplicación de gestión de workflows desarrollada por la compañía OrangeScape Inc.

Se ejecuta como *SaaS* en la nube a través de interfaz web.

3.5.1 Diseñador

KissFlow dispone de una herramienta gráfica con la que se pueden definir los workflows. Esta herramienta no sigue el estándar BPMN y modela los workflows de una manera mucho más simple, como una secuencia de acciones que tienen un determinado responsable. Se destaca que se puede asignar como responsable de una acción al "jefe" de un determinado usuario. Esto resulta útil para modelar workflows de tipo solicitud y aprobación.

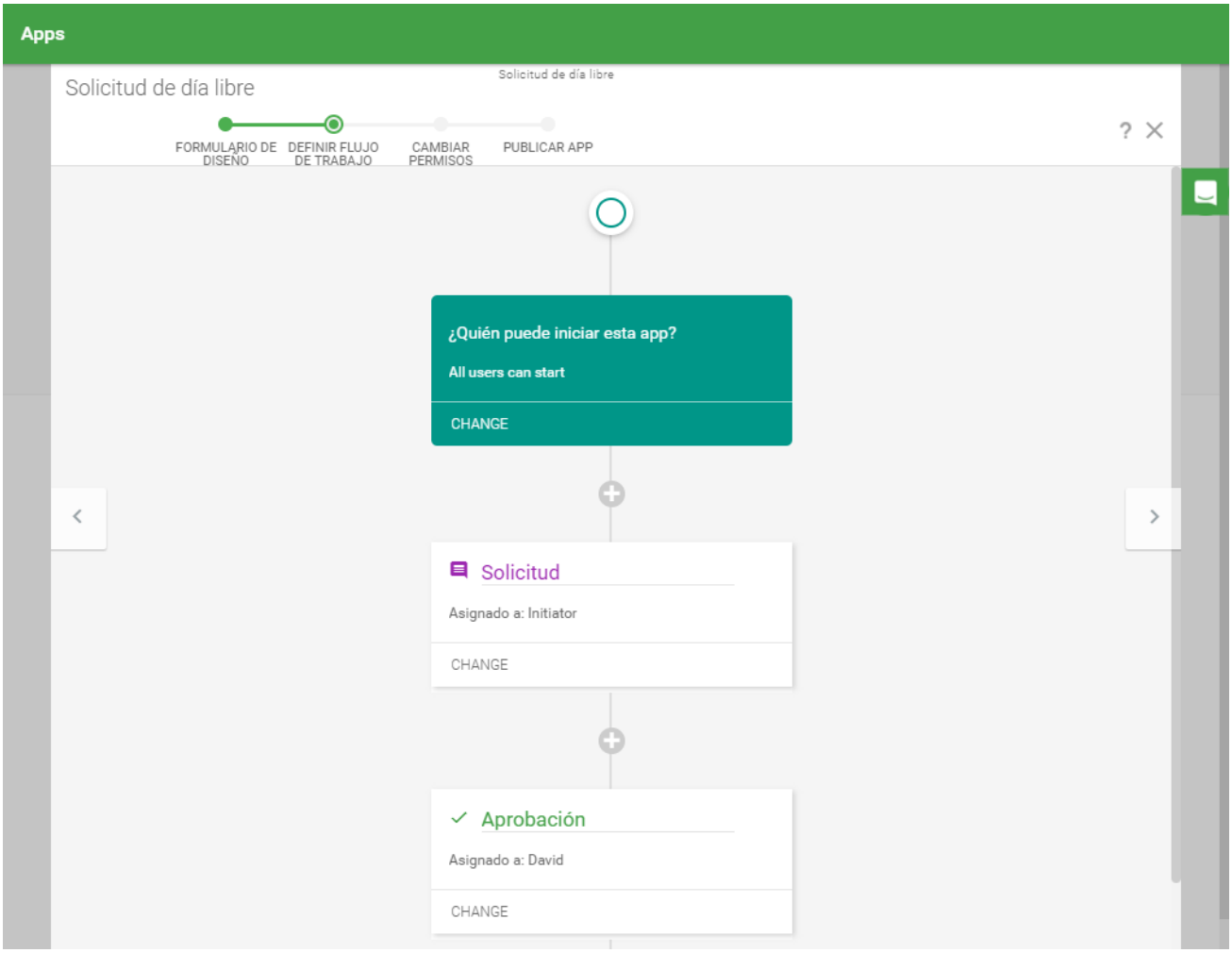


Figura 24: Diseñador de workflows de KissFlow

KissFlow permite asociar formularios a las acciones para mostrar y recibir información del usuario. El diseñador de formularios es muy básico y no permite realizar configuraciones visuales, tan sólo definir los campos que se mostrarán en una columna uno encima de otro. Se destaca que tiene la posibilidad de marcar los campos como "obligatorios", y también la posibilidad de aplicar reglas de "validación" al valor introducido por el usuario.

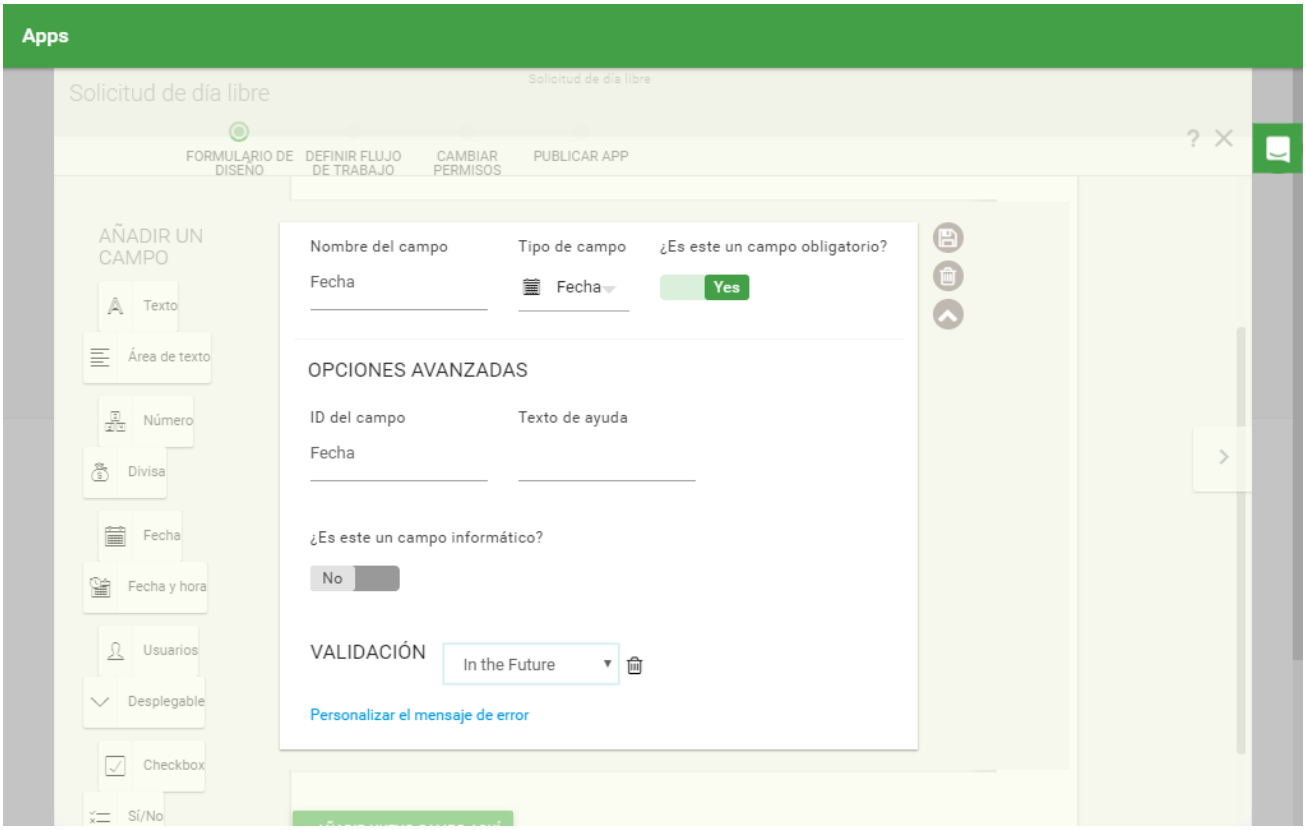


Figura 25: Diseñador de formularios de KissFlow

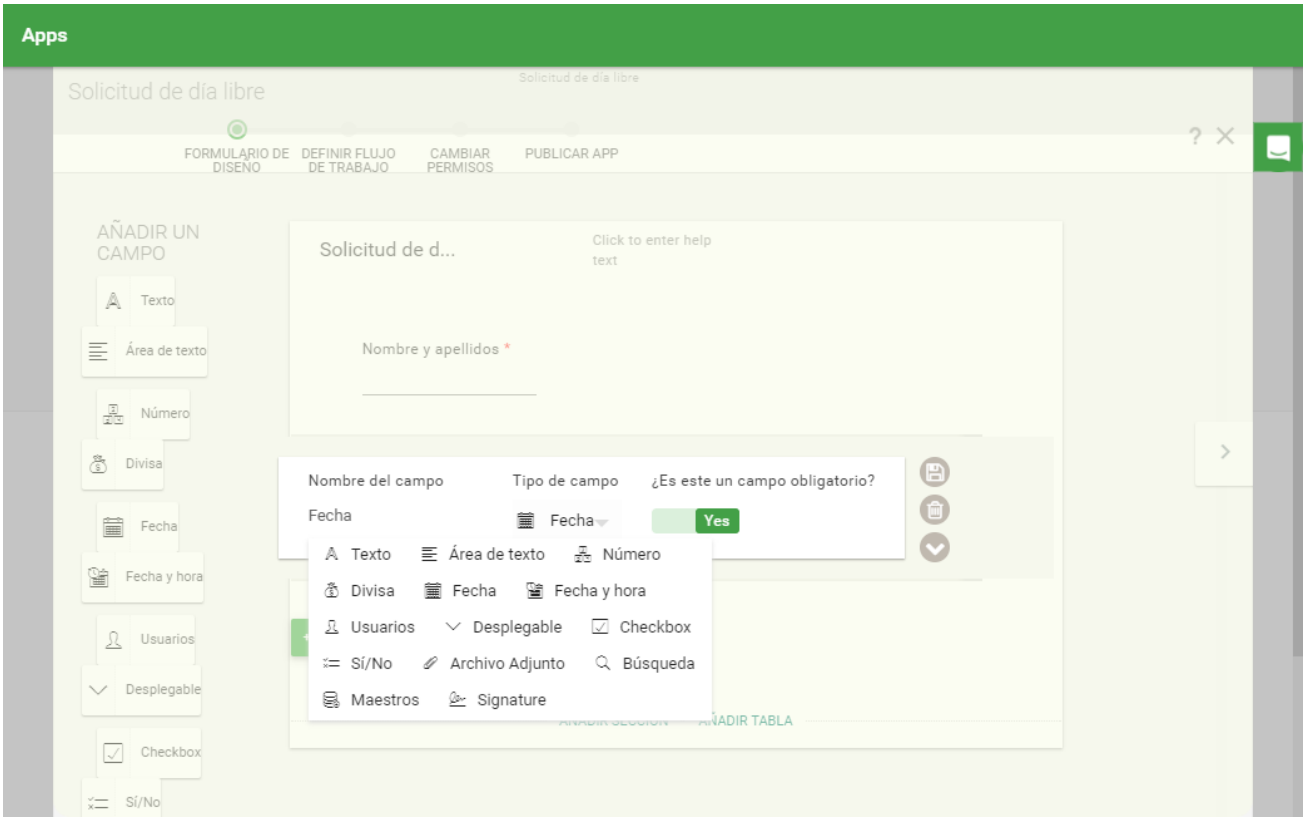


Figura 26: Diseñador de formularios de KissFlow

3.5.2 Interfaz de usuario

La interfaz con el usuario tiene un estilo muy actual y resulta muy intuitiva. Además, está disponible en multi-idioma. Permite al usuario iniciar nuevos workflows, listar los que tiene pendientes y poder acceder a ellos abriendo su formulario. Una característica a destacar es que permite que el usuario "delegue" en otro, o sea, que el usuario puede transferir sus workflows a otro usuario durante un periodo de tiempo definido. Esta funcionalidad resulta muy útil para que un usuario pueda delegar sus tareas durante su ausencia.

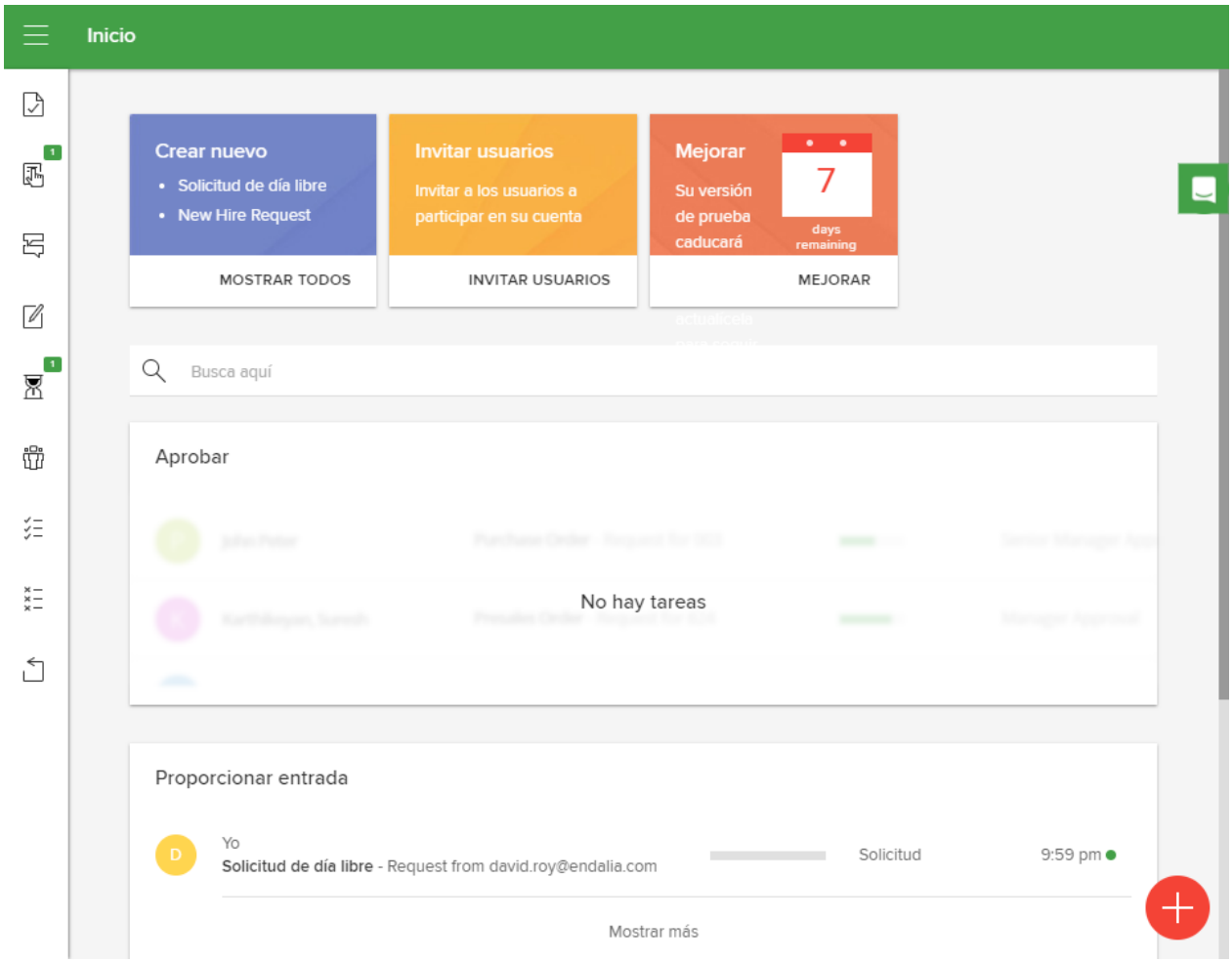


Figura 27: Pantalla principal de usuario de KissFlow

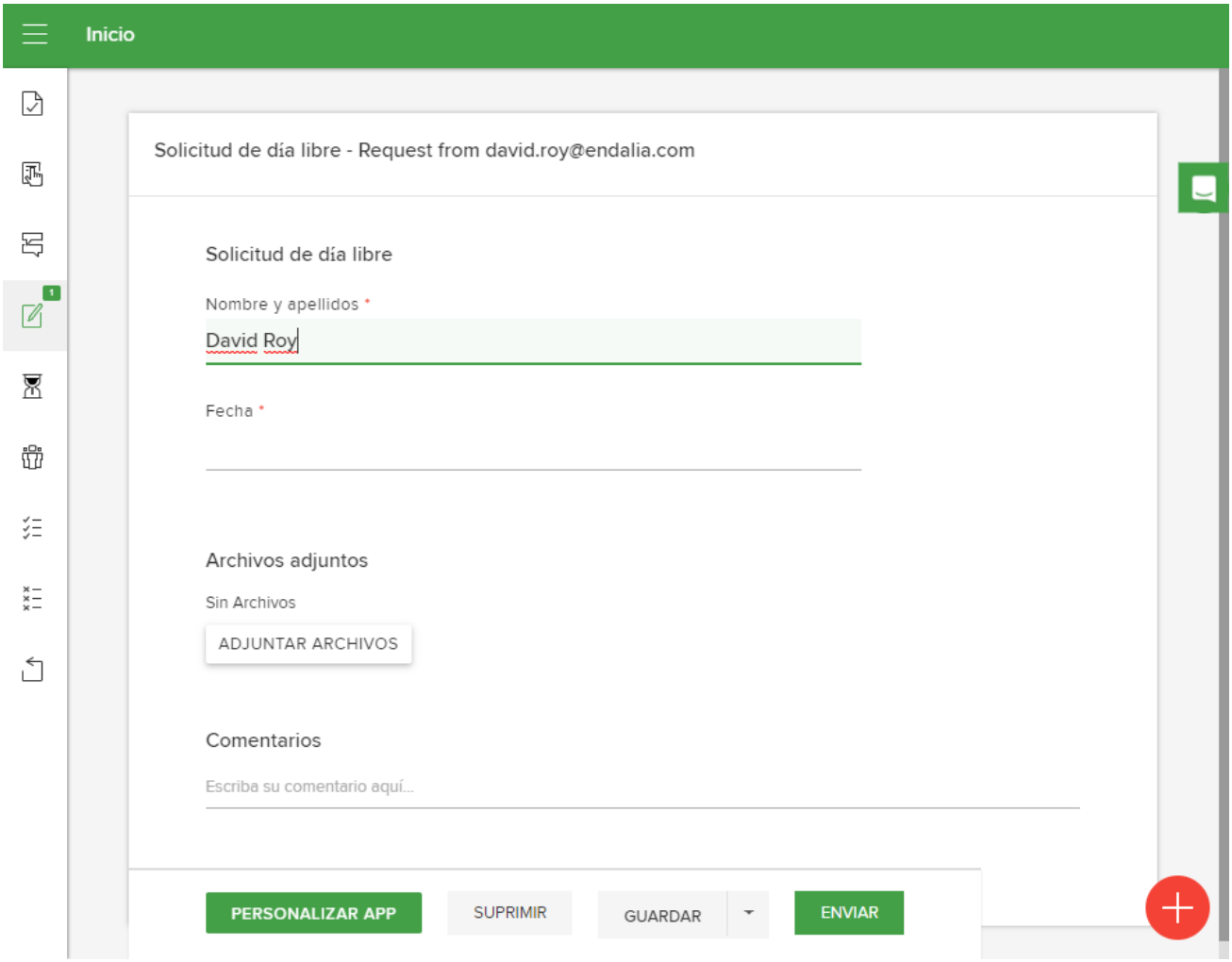


Figura 28: Pantalla de usuario al abrir una instancia

3.5.3 Interconexiones

KissFlow permite conectarse con un servidor de correo para poder enviar notificaciones de email.

La interconexión con otros sistemas sólo se puede realizar a través de *webhooks*[2].

[2] *Webhook*: una llamada HTTP a una determinada url que está preparada para desencadenar una determinada acción al recibir una conexión.

3.6 Conclusiones

Tras el estudio de las herramientas existentes en el mercado se destacan las características que se han considerado más importantes y que se añadirán a los requisitos del sistema:

- Los workflows estarán categorizados o agrupados por familias.
- Los workflows se podrán poner en estado *histórico* o *desactivado*. Con esto no se podrán iniciar nuevos workflows, pero no se borrarán los ya existentes.
- El usuario podrá consultar sus workflows filtrándolos por diferentes criterios.
- Los workflows tendrán soporte multi-idioma.
- El usuario podrá ver un diagrama o esquema del workflow que le ayude a entender cómo se relacionan las distintas fases del flujo.
- Cada fase del workflow tiene asociado un formulario para mostrar y recoger la información del usuario.
- Los formularios estarán compuestos por campos que tendrán un nombre y un valor.
- Los valores de estos campos estarán tipados (texto, número entero, fecha, etc)
- Los campos podrán ser obligatorios.
- Los campos podrán ser de sólo lectura.
- Los campos podrán ocultarse según determinadas condiciones.
- Se podrán realizar validaciones en los valores introducidos por el usuario.
- Eventos: antes y/o después de cada fase, se podrán ejecutar determinadas acciones en la lógica de negocio, en este caso, EndaliaHR.
- Eventos: antes y/o después de cada fase, se podrán enviar emails de notificación.
- Los responsables de ejecutar cada fase del workflow se podrán asignar de manera fija o de manera dinámica (teniendo en cuenta unas determinadas condiciones del workflow, e.g. “el jefe de”).
- Los responsables de las fases podrán ser usuarios individuales o grupos de usuarios determinados por un permiso o privilegio (EndaliaHR ya posee un sistea de usuarios y permisos).
- Se creará el rol *supervisor* de workflows que permitirá visualizar cualquier workflow aunque no sea responsable explícito de la fase.
- Los workflows se podrán delegar en otro usuario durante un periodo de tiempo determinado (a tener en cuenta en futuras ampliaciones del sistema).

4. BIBLIOGRAFÍA

[1] Wikipedia, «BPMN,» [En línea]. Available: https://es.wikipedia.org/wiki/Business_Process_Model_and_Notation.

[2] «XPDL,» [En línea]. Available: http://www.xpdl.org/.

[3] «WfMC,» [En línea]. Available: http://wfmc.org/.

[4] «BPEL,» [En línea]. Available: https://es.wikipedia.org/wiki/WS-BPEL.

[5] «jBPM,» [En línea]. Available: https://www.jbpm.org/.

[6] «Bonita BPM,» [En línea]. Available: http://www.bonitasoft.com/products.

[7] «ProcessMaker,» [En línea]. Available: https://www.processmaker.com/es.

[8] «WorkflowGen,» [En línea]. Available: https://www.workflowgen.com/.

[9] «KissFlow,» [En línea]. Available: https://kissflow.com/.

[10] «Activiti,» [En línea]. Available: https://www.activiti.org/.

[11] «IBM WPS,» [En línea]. Available: https://www-01.ibm.com/software/integration/wps/.

[12] «Windows Workflow Foundation,» [En línea]. Available: https://msdn.microsoft.com/es-es/library/dd489441(v=vs.110).aspx.

[13] «YAWL,» [En línea]. Available: http://www.yawlfoundation.org/.

[14] «Oracle BPEL Process Manager,» [En línea]. Available: http://www.oracle.com/technetwork/middleware/bpel/overview/index.html.

[15] «SAP Business Workflow,» [En línea]. Available: https://archive.sap.com/documents/docs/DOC-31056.

[16] Krasimira P. Stoilova, Todor A. Stoilov, «Evolution of the workflow management systems,» [En línea]. Available: https://pdfs.semanticscholar.org/7fe7/38eb61dcdd27d7b59cddc1c929c85326c270.pdf.

[17] Michael zur Muehlen, Workflow-based Process Controlling. Foundation, Design and Application of Workflow-driven Process Information Systems, http://www.cebpi.org/wp-content/uploads/2011/05/Michael_zur_Muehlen_-_Workflow-based_Process_Controlling_Web.pdf: Logos Verlag Berlin, ISBN 3-8325-0388-9, 2004.

[18] Wikipedia, «XPDL,» [En línea]. Available: https://es.wikipedia.org/wiki/XPDL.

[19] Denis Gagné, Simon Ringuette, «BPMN Quick Guide, 2nd edition,» [En línea]. Available: http://www.bpmnquickguide.com/view-bpmn-quick-guide/.

[20] Oasis, «WS-BPEL,» [En línea]. Available: http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf.

[21] «Anexo Documento Visión, sección 2.3».

[22] J. César, «Modelado de negocios: BPMN,» [En línea]. Available: http://www.juliocesar.in/2012/09/modelado-de-negocios-bpmn-business.html.

[23] J. Camacho, «BPMN: Ejemplo servicio de pizzería,» [En línea]. Available: http://teccompeinfo.blogspot.com.es/2013/05/bpmn-ejemplo-servicio-pizzeria-con-la.html.

endalia®
HR Consulting & Software

III. MODELO DE NEGOCIO

WORKFLOW MANAGEMENT SYSTEM

DAVID ROY

TABLA DE CONTENIDOS

Tabla de contenidos 2

1. Introducción 3

1.1 Objetivo 3

1.2 Alcance 3

1.3 Organización del documento 3

2. Casos de uso 3

2.1 Casos de uso principales 3

2.2 Caso de uso: Consultar instancias 5

2.3 Caso de uso: avanzar instancia 5

2.4 Caso de uso: retroceder instancia 6

2.5 Caso de uso: cancelar instancia 6

3. Prototipos GUI 7

3.1 Pantalla maestro 8

3.2 Pantalla detalle 9

4. Modelo de negocio 11

4.1 Workflow 11

4.2 Step 12

4.3 Action 12

4.4 Notification 12

4.5 User 13

4.6 Field 13

4.7 Instances 13

4.8 Conjunto completo 14

5. Bibliografía 14

1. INTRODUCCIÓN

1.1 Objetivo

El objetivo de este documento es identificar los principales casos de uso del sistema a desarrollar. En la metodología RUP [1] los casos de uso son un elemento fundamental que sirve de guía a lo largo de las distintas fases del proyecto.

1.2 Alcance

El alcance de este documento abarca las fases de inicio y elaboración, durante las cuales los casos de uso se van ampliando y refinando conforme el desarrollo avanza.

1.3 Organización del documento

El presente documento contiene las siguientes secciones:

1. Introducción: Descripción del objetivo y alcance del documento.
2. Casos de uso.
3. Prototipos GUI.
4. Modelo de negocio.
5. Bibliografía: Referencias a libros y webs utilizadas en el documento.

2. CASOS DE USO

El modelo de negocio es una técnica de análisis especificada en RUP y utilizada para comprender los procesos de negocio de la organización: "*Describe los procesos de negocio de una organización en términos de casos de uso del negocio y actores del negocio, que corresponden con los procesos del negocio y los clientes respectivamente*" [2]

Los casos de uso tienen por finalidad identificar quién va a usar el sistema y para qué lo va a usar. Además, los casos de uso brindan una oportunidad para encontrar nuevos requisitos funcionales. Los requisitos no funcionales no se plasman en los casos de uso. De igual modo, los casos de uso no deben utilizarse para hacer una descomposición funcional del sistema.

2.1 Casos de uso principales

Durante el presente estudio, se identifican 3 tipos de usuarios para el sistema, que son los actores:

- Usuario del WfMS: usuario de la aplicación.
- Participante de workflows: un usuario cuando es participante/responsable de un paso.
- Administrador de workflows.

Desde la perspectiva de estos actores se localizan las funcionalidades del sistema que tienen importancia para ellos:

- Usuario:
    - Consultar workflows: localizar aquellos workflows en los que es participante, listar cuántos workflows ha realizado durante un determinado periodo de tiempo, etc.
- Participante:
    - Ejecutar workflow: es el caso de uso principal.
- Administrador:
    - Gestionar workflows: crear, editar y borrar las plantillas de workflows.
    - Gestionar los participantes: para gestionar los participantes de los workflows, es necesario gestionar los usuarios y permisos del sistema en el que se integran, que es EndaliaHR. Por esta razón este caso de uso queda fuera del sistema a desarrollar.

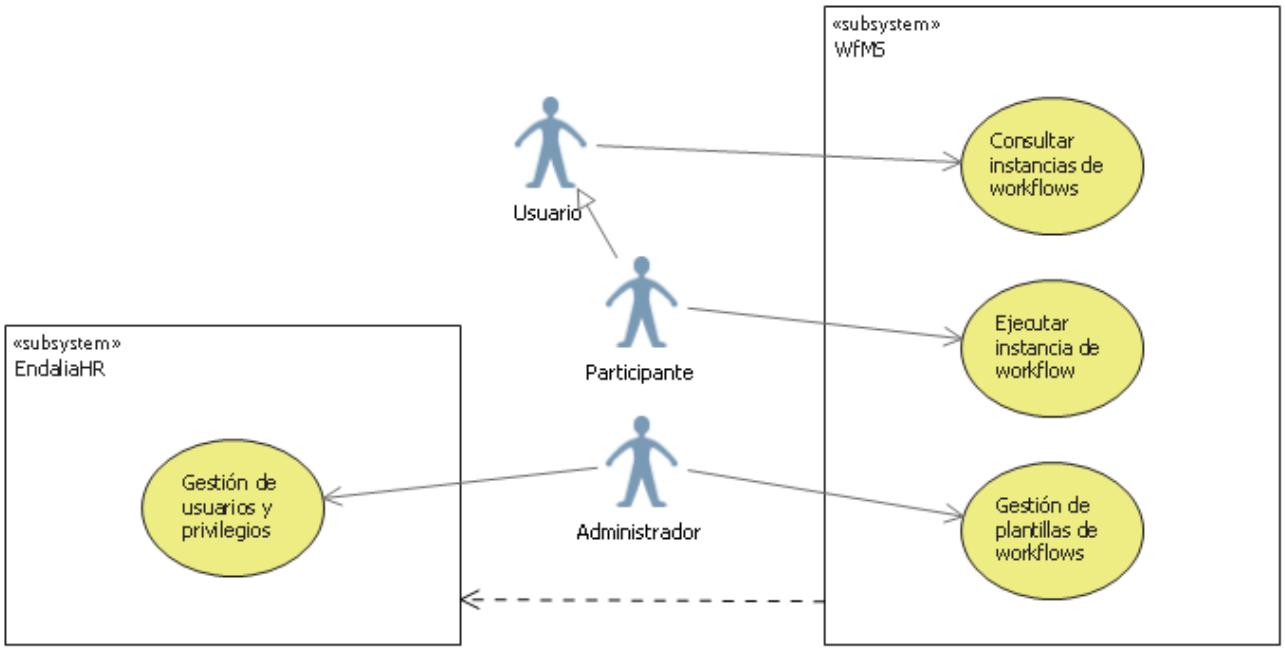


Figura 1: Diagrama de casos de uso principales

A continuación, se van a detallar individualmente cada uno de estos casos de uso. En el caso de uso de "Gestión de plantillas de workflows", al tratarse de funcionalidad CRUD[1], no se considera necesario detallarlo más profundamente.

[1] CRUD: *Create, Read, Update, Delete*. Se refiere las operaciones habituales que se pueden realizar sobre una determinada entidad del sistema.

2.2 Caso de uso: Consultar instancias

A partir de este caso de uso, se identifican 2 casos de uso más específicos:

- Filtrar instancias activas: aquellas que están pendientes de realización por parte del participante
- Filtrar instancias finalizadas: aquellas que el participante ya ha realizado.

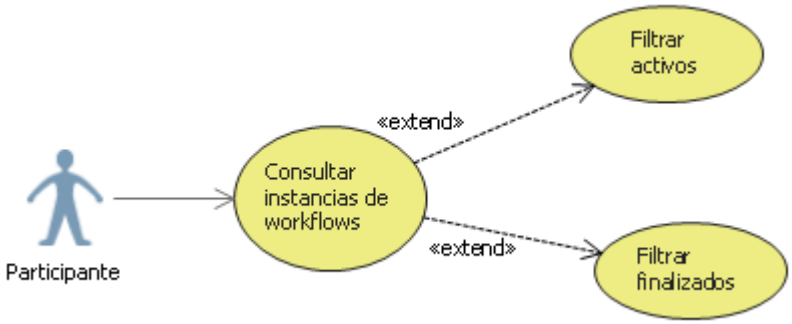


Figura 2: Diagrama de caso de uso de consulta de instancias

2.3 Caso de uso: avanzar instancia

A partir del caso de uso general “Ejecutar instancia” se identifican otros 3:

- Avanzar la instancia: realizar la tarea asignada y avanzar el flujo de trabajo al siguiente paso.
- Retroceder la instancia: el participante no puede realizar la tarea asignada por algún motivo y devuelve el flujo de trabajo al paso anterior para que solucionen el problema y lo vuelvan a avanzar a su fase.
- Cancelar la instancia: el participante detecta que el proceso no puede continuar y decide abortarlo por completo (diferente de retroceder a una fase anterior).

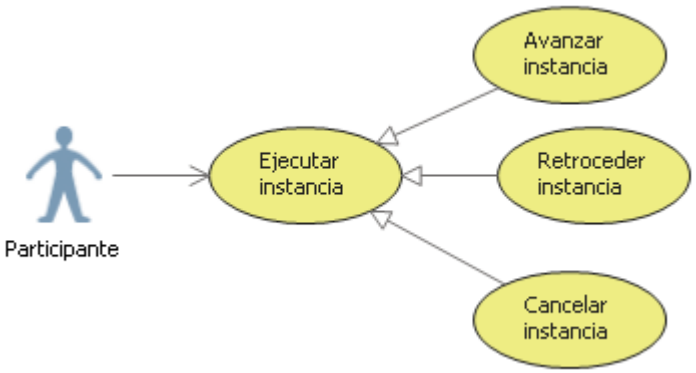


Figura 3: Diagrama de caso de uso Ejecutar instancia

El caso de uso avanzar la instancia, se puede representar con el siguiente diagrama:

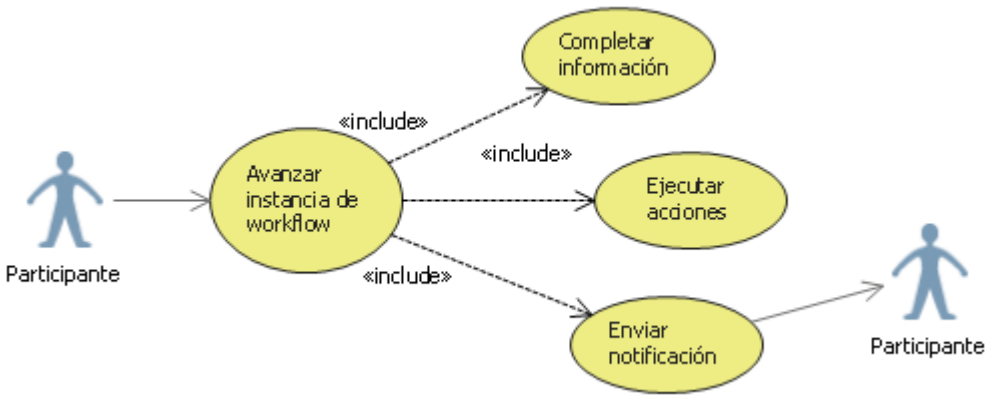


Figura 4: Diagrama de caso de uso de avanzar instancia

Avanzar al siguiente paso implica que el participante completa los campos de información solicitados en el workflow, se ejecutan las acciones programadas para dicho paso y se envían las notificaciones al resto de participantes.

2.4 Caso de uso: retroceder instancia

Al definir este caso de uso, se encuentra una funcionalidad que cabe destacar y que hay que anotar en el documento de requisitos funcionales: cuando una instancia retrocede al paso anterior se deben realizar ciertas acciones. De igual modo que al avanzar de fase se ejecutan ciertas acciones, al retroceder al paso anterior, puede tener sentido tener que *deshacer* esas acciones. Esto se ha reflejado en el diagrama del caso de uso:

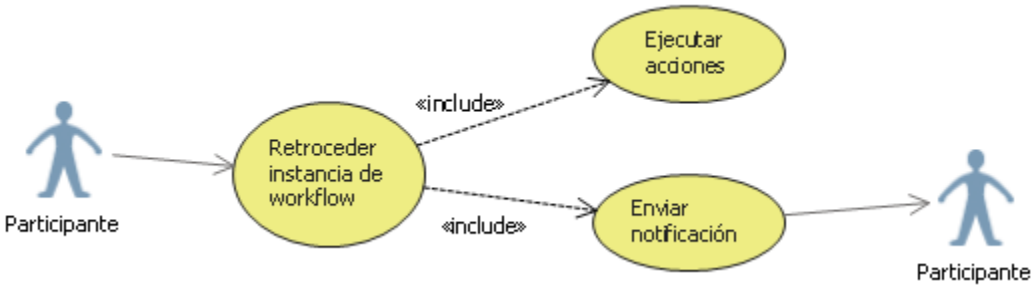


Figura 5: Caso de uso de retroceder instancia

2.5 Caso de uso: cancelar instancia

En este caso de uso, cuando el usuario decide cancelar o abortar el workflow, se debe notificar a los participantes implicados tal circunstancia.

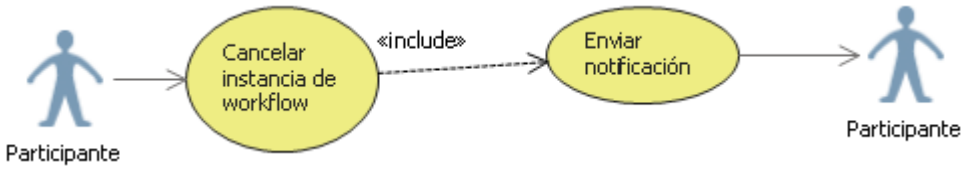


Figura 6: Caso de uso de cancelar instancia

3. PROTOTIPOS GUI

Los prototipos de interfaz gráfica de usuario (GUI) sirven como complemento para identificar los casos de uso del sistema y ayudan en la fase de recogida de requisitos con el cliente.

Como se ha indicado en los objetivos del proyecto, el sistema que se está desarrollando debe integrarse en el entorno web de EndaliaHR y por tanto los prototipos GUI se han preparado teniendo en cuenta el aspecto visual de EndaliaHR. A continuación, se muestra un prototipo con los elementos fijos de EndaliaHR (menú lateral y cabecera) y con el área principal, en blanco, que es donde se integrará el proyecto:

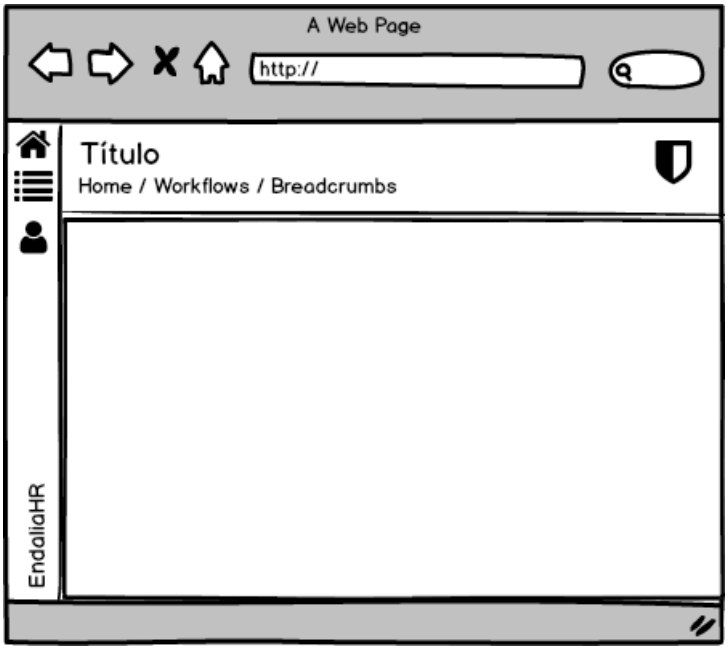


Figura 7: Elementos fijos del GUI

El menú lateral de EndaliaHR cuenta con 3 botones principales:

- Inicio: nos lleva directamente a la página de inicio.
- Menú: despliega el menú con las distintas pantallas del sistema, organizadas por módulos.
- Opciones de usuario: permite, entre otros, cambiar el idioma, cerrar la sesión, etc.

La cabecera de EndaliaHR muestra el título de la página y debajo la miga de pan o *breadcrumb* [3] (una ayuda visual para el usuario que le indica la secuencia de páginas que ha visitado). En la parte superior derecha se muestra el logotipo del cliente.

Para este proyecto los prototipos que se han preparado siguiendo el patrón *maestro-detalle* [4], con una pantalla *maestro* que listará los workflows disponibles y desde ella se podrá acceder a la pantalla *detalle* con la información completa de cada workflow por separado.

3.1 Pantalla maestro

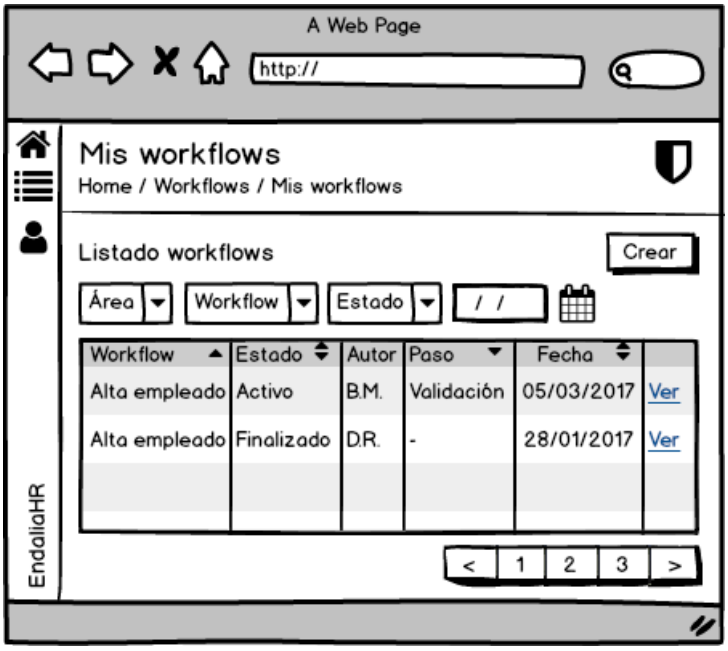


Figura 8: Pantalla maestro

En la parte superior se encuentran los filtros y un botón para crear nuevas instancias de workflow, en la parte central se encuentra el listado de workflows y en la parte inferior se muestra el paginado de la lista.

Nótese que la última columna de cada fila da acceso a la vista detalle, que se explicará en la siguiente sección.

Se ha añadido el concepto de "área", que permite que los workflows estén organizados por temáticas y por tanto ayuda a que el GUI sea más intuitivo y cómodo para el usuario. A continuación, se muestra el prototipo de la pantalla tras pulsar el botón "Crear", donde se pone de manifiesto la utilidad de organizar los workflows en áreas:

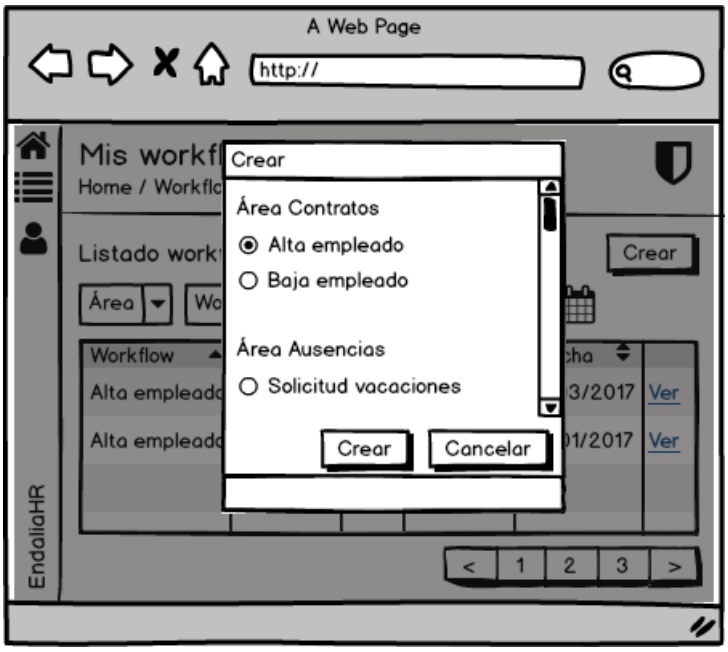


Figura 9: Prototipo tras pulsar el botón Crear

3.2 Pantalla detalle

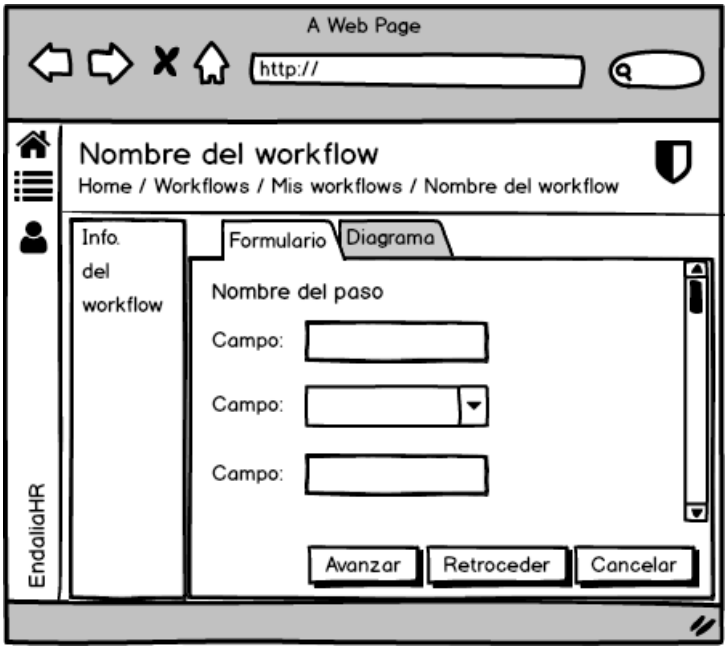


Figura 10: Prototipo de pantalla detalle

En la pantalla detalle se muestra la información completa del workflow:

- Banda izquierda: datos sobre el proceso, fechas, autor, etc.
- Región principal: tiene 2 pestañas:
    - Formulario: muestra los campos del paso, uno encima de otro.
    - Diagrama: muestra visualmente un esquema de los pasos del workflow.
- Región inferior: los botones de acción que permiten avanzar o retroceder el workflow.

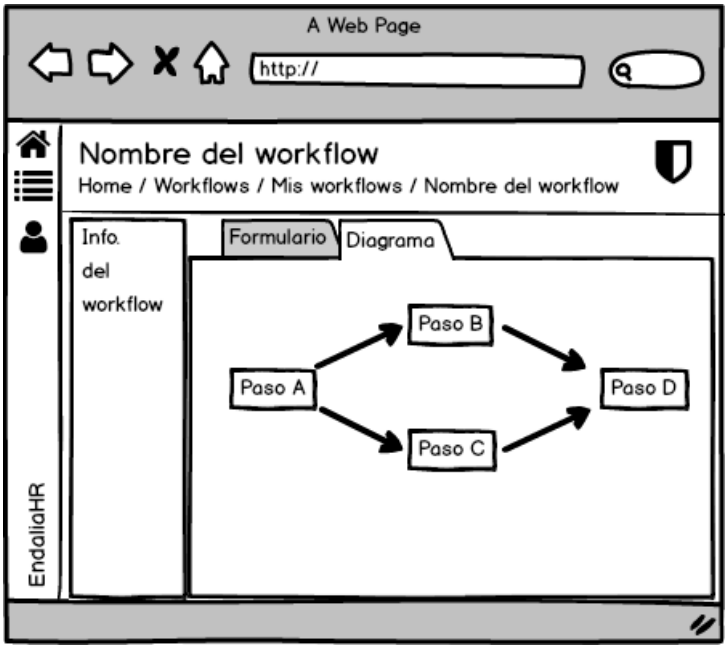


Figura 11: Pantalla detalle, pestaña Diagrama

Prestando atención a los campos del formulario, éstos pueden ser de diferentes tipos:

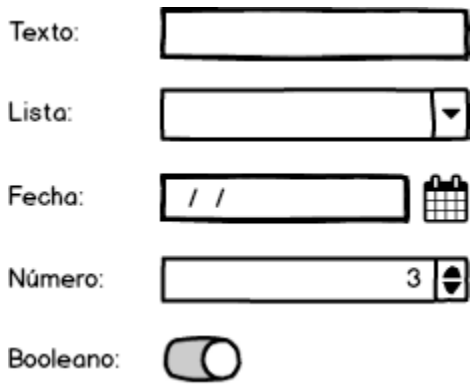


Figura 12: Prototipos de tipos de campo

Para mejorar la experiencia de uso, los campos tendrán, además del nombre, una descripción opcional que servirá de ayuda al usuario a la hora de rellenar el campo. Para los casos en que el texto de la descripción no sea suficiente, se mostrará un botón de información adicional que ofrecerá al usuario una ayuda más extensa. Todos estos detalles se pueden apreciar en el siguiente prototipo:

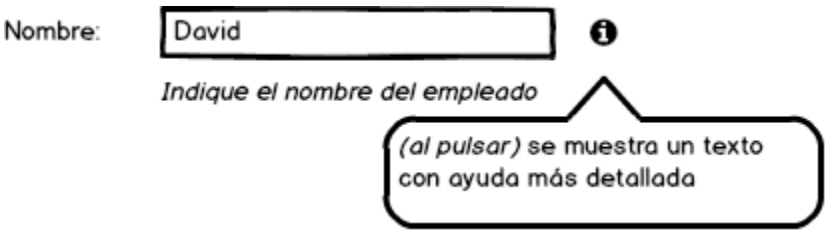


Figura 13: Prototipo de campo

4. MODELO DE NEGOCIO

Como parte de este análisis, se va a definir un modelo conceptual de las clases que componen el sistema. En los siguientes apartados se explica el modelo por partes y al final de la sección se muestra el conjunto completo.

Se han aplicado los estándares de codificación definidos en el Plan de proyecto y las entidades se han nombrado en inglés.

4.1 Workflow

Figura 14: Modelo conceptual de workflow

El *ExecutionOrder* relaciona cada paso con 0 o más pasos siguientes. A su vez un paso puede tener 0 o más pasos inicio. Este modelo conceptual permite representar flujos no lineales:

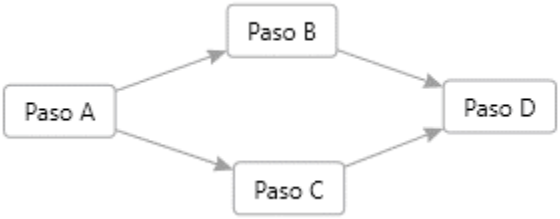


Figura 15: Esquema de flujo no lineal

Para el ejemplo de la figura 3, el *ExecutionOrder* almacenaría las siguientes relaciones:

StepInit	StepNext
A	B
A	C
B	D
C	D

4.2 Step

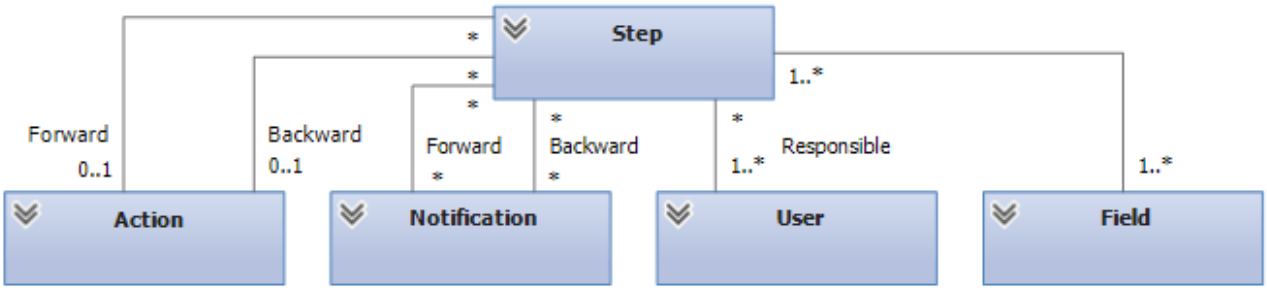


Figura 16: Modelo conceptual Paso

Un paso tiene:

- Acción al avanzar: es opcional.
- Acción al retroceder: es opcional.
- Notificaciones al avanzar: cero o más notificaciones que se envían al avanzar al siguiente(s) paso(s).
- Notificaciones al retroceder: cero o más notificaciones.
- Responsable: 1 o más responsables de ejecutar este paso.
- Campos: 1 o más campos de información.

4.3 Action

Una acción es la abstracción de una llamada al sistema con la lógica de negocio de la compañía para que ejecute unas determinadas funciones. En este caso, representan las funciones que se ejecutan en el sistema EndaliaHR.

4.4 Notification

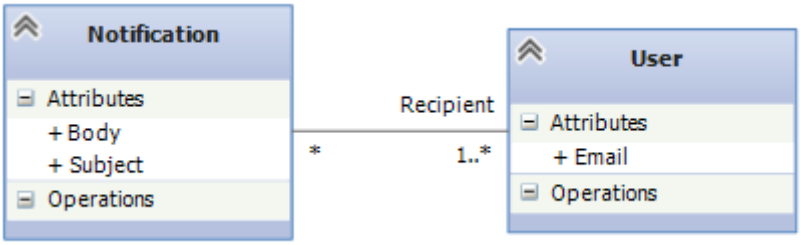


Figura 17: Modelo conceptual Notificación

La notificación tiene:

- Asunto del mensaje enviado.
- Cuerpo del mensaje enviado.
- Destinatario: 1 o más destinatarios, con dirección de email definida, a los que se les enviará la notificación.

4.5 User

La clase usuario queda reflejada en el modelo de negocio a modo informativo, pero es una clase que se integra desde el sistema EndaliaHR.

4.6 Field

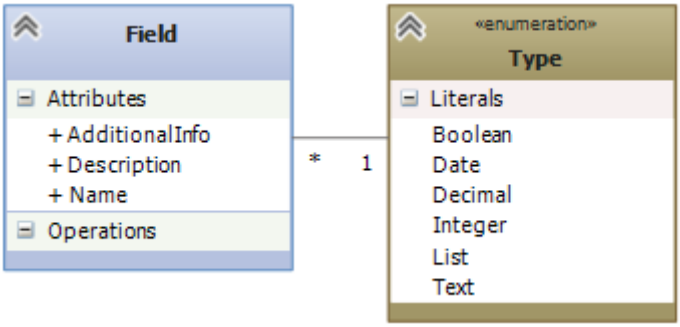


Figura 18: Modelo conceptual Campo

Un campo tiene:

- Nombre.
- Descripción.
- Información adicional
- Tipo. Las posibles tipologías de un campo se pueden ampliar para cumplir nuevos requerimientos.

4.7 Instances

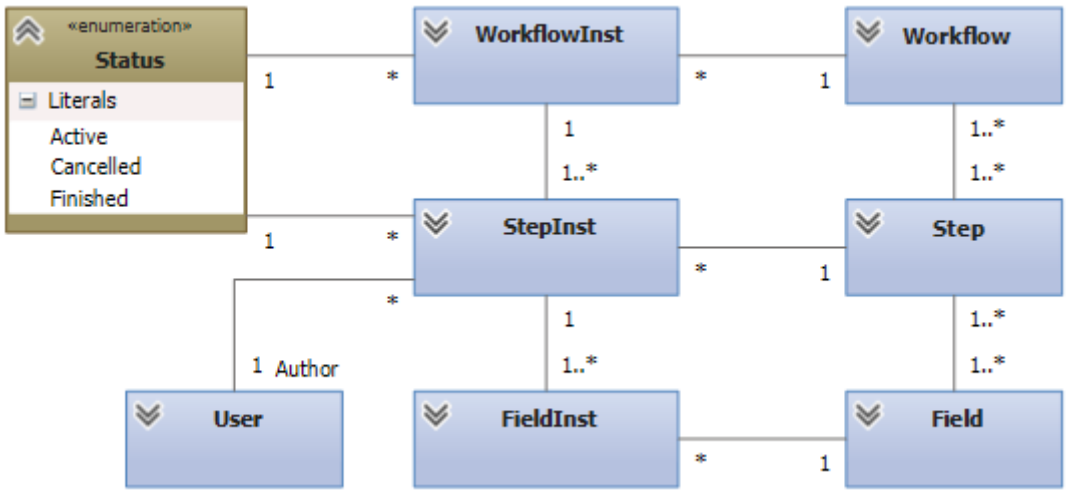


Figura 19: Modelo conceptual de instancias

Las instancias de workflows se modelan con 3 clases:

- Instancia de workflow.
- Instancia de paso.
- Instancia de campo.

Cada una de ellas representa una ejecución del workflow y entre otras cosas almacenan el estado y el autor del paso.

4.8 Conjunto completo

A continuación, se muestra en conjunto todas las entidades y relaciones identificadas anteriormente:

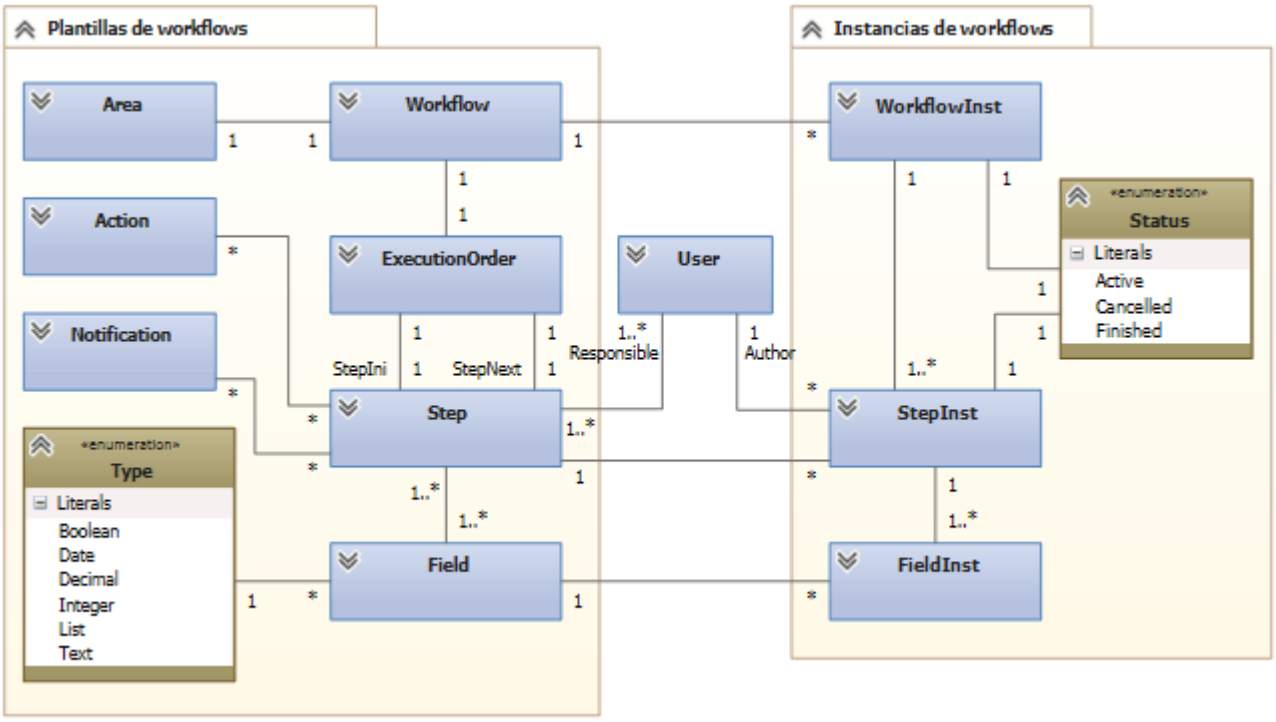


Figura 20: Modelo conceptual completo

5. BIBLIOGRAFÍA

[1] «RUP,» [En línea]. Available: https://es.wikipedia.org/wiki/Proceso_Unificado_Racional.

[2] I. Jacobson, G. Booch, J. Rumbaugh, El Proceso Unificado de Desarrollo de Software, Pearson Education, 2000.

[3] Wikipedia, «Breadcrumb (Miga de pan),» [En línea]. Available: https://es.wikipedia.org/wiki/Miga_de_pan_(inform%C3%A1tica).

[4] Wikipedia, «Master-Detail interface,» [En línea]. Available: https://en.wikipedia.org/wiki/Master%E2%80%93detail_interface.

endalia®
HR Consulting & Software

IV. ESPECIFICACIÓN DE REQUISITOS

WORKFLOW MANAGEMENT SYSTEM

DAVID ROY

TABLA DE CONTENIDOS

Tabla de contenidos 2
1. Introducción 3
1.1 Objetivo 3
1.2 Alcance 3
1.3 Organización del documento 3
2. Requisitos del Sistema 3
2.1 Requisitos funcionales 4
2.2 Requisitos no funcionales 7
3. Bibliografía 8

1. INTRODUCCIÓN

1.1 Objetivo

El presente documento pretende identificar y describir los requisitos del sistema a desarrollar. Por requisitos se entienden las características y funcionalidades que debe cumplir el nuevo sistema.

1.2 Alcance

Durante la fase inicial del proyecto se completarán unos requisitos iniciales que se irán ampliando y aumentando durante la fase de elaboración.

1.3 Organización del documento

El presente documento contiene las siguientes secciones:

1. Introducción: Descripción del objetivo y alcance del documento.
2. Requisitos del sistema: Se muestra la relación de los requisitos del sistema a desarrollar, su descripción y su nivel de criticidad, diferenciando los requisitos funcionales de los no funcionales.
3. Bibliografía: Referencias a libros y webs utilizadas en el documento.

2. REQUISITOS DEL SISTEMA

En esta sección se van a detallar los diferentes requisitos que el sistema a desarrollar debe satisfacer. Una buena especificación de requisitos debe ser [1]:

- Correcta
- Inequívoca
- Completa
- Consistente
- Comprobable
- Modificable
- Identificable

Los requisitos van a estar divididos en dos grupos:

- Requisitos funcionales. Describe las funcionalidades del sistema, referidas a:
    - Funciones de consulta y actualización de datos.
    - Datos manejados.
    - Interacción con otros sistemas.
- Requisitos no funcionales. Describen las facilidades que debe proporcionar el sistema, referidas a:
    - Rendimiento.
    - Volumen y tamaño de datos.
    - Frecuencia de tratamiento.
    - Requisitos de seguridad:
        - Control de accesos.

        - Integridad de la información.
    - Requisitos especiales de comunicaciones.

2.1 Requisitos funcionales

Nombre	RF – 01: Gestión de definiciones de workflows
Descripción	El WfMS permitirá al administrador añadir, editar y borrar definiciones de workflows

Nombre	RF – 02: Gestión de usuarios y privilegios
Descripción	Los administradores pueden crear, editar y borrar usuarios en el sistema, asignar permisos a usuarios y controlar si participan en workflows o no. Este requisito recae sobre el sistema EndaliaHR en el que se va a integrar el WfMS.

Nombre	RF – 03: Consultar workflows según criterios
Descripción	El sistema permite consultas de las instancias de workflows filtrando por: • Área (ver RF-14). • Estado. • Tipo de workflow. • Fecha de ejecución. • Autor. • Responsable.

Nombre	RF – 04: Notificaciones por email
Descripción	El sistema permite enviar notificaciones por email a los participantes, con información acerca del workflow. El tipo de notificación, los destinatarios y la información enviada estará indicada en la plantilla del workflow.

Nombre	RF – 05: Múltiples participantes por cada paso
Descripción	Los pasos pueden tener 1 o más participantes y cualquiera de ellos puede ejecutar el workflow y ver la información del paso.

Nombre	RF – 06: Participantes indicados explícitamente
Descripción	El sistema permite indicar los participantes de un paso de forma explícita, indicando unos usuarios en concreto.

Nombre	RF – 07: Participantes indicados colectivamente
Descripción	El sistema permite indicar los participantes de un paso colectivamente, indicando un privilegio. Aquellos usuarios que tengan ese privilegio concedido serán participantes.

Nombre	RF – 08: Participantes indicados selectivamente
Descripción	El sistema permite indicar los participantes de un paso de forma selectiva, mediante una condición. Esta condición estará indicada en la plantilla del workflow. E.g: los usuarios activos.

Nombre	RF – 09: Instancias de paso simultáneas
Descripción	Puede haber varios pasos de una misma instancia activos al mismo tiempo. Puede haber workflows no lineales.

Nombre	RF – 10: Posibilidad de retroceder un paso
Descripción	El participante de un workflow puede retroceder al paso inmediatamente anterior (si existe). Esta opción puede deshabilitarse en la plantilla del workflow

Nombre	RF – 11: Posibilidad de cancelar una instancia
Descripción	El participante de un workflow puede cancelar o abortar la instancia del workflow en cualquier paso. Esta opción puede deshabilitarse en la plantilla del workflow

Nombre	RF – 12: Al avanzar de paso se ejecutan acciones
Descripción	En la plantilla de workflow se pueden configurar acciones que se ejecutan automáticamente al avanzar al paso siguiente. Son opcionales.

Nombre	RF – 13: Al retroceder de paso se ejecutan acciones
Descripción	En la plantilla de workflow se pueden configurar acciones que se ejecutan automáticamente al retroceder al paso anterior. Son opcionales.

Nombre	RF – 14: Los workflows se agrupan en áreas
Descripción	Los workflows se pueden clasificar en áreas.

Nombre	RF – 15: Los workflows tienen pueden pasar a estado histórico
Descripción	Los workflows en estado histórico no permitirán crear nuevas instancias. No afectará a las instancias ya creadas.

Nombre	RF – 16: Los workflows podrán definirse en varios idiomas
Descripción	Los workflows tendrán soporte para mostrar los mensajes por pantalla y las notificaciones de email en diferentes idiomas.

Nombre	RF – 17: El usuario podrá visualizar un diagrama del workflow
Descripción	El usuario tendrá la opción de visualizar gráficamente las fases del workflow, las fases activas y las flechas que indican la secuencia de las fases.

Nombre	RF – 18: Los campos de los workflows tendrán un tipo de valor.
Descripción	Los campos guardarán el valor en un determinado tipo. Los tipos disponibles serán: • Texto • Entero • Decimal • Moneda (valor decimal, con 2 decimales y acompañados por el símbolo de la moneda) • Fecha • Nº de cuenta corriente (incluyendo el Iban) • Selección simple de entre una lista de valores. • Selección múltiple de una lista. • Selección de un empleado

- Selección de un puesto de trabajo.

Nombre	RF – 19: Los campos podrán ser obligatorios
Descripción	No se permitirá avanzar de fase si hay algún campo obligatorio que no ha sido rellenado.

Nombre	RF – 20: Los campos podrán ser de sólo lectura
Descripción	El usuario podrá visualizar el valor, pero no modificarlo.

Nombre	RF – 21: Los campos podrán ocultarse
Descripción	En la plantilla del workflow se podrán definir condiciones que indiquen si un determinado campo debe ocultarse al usuario. El valor podrá ser tenido en cuenta para otras operaciones de la aplicación, pero el usuario no podrá visualizarlo.

Nombre	RF – 22: Los campos podrán validarse.
Descripción	En la plantilla del workflow podrán definirse reglas de validación de campos, de modo que no se permita avanzar de fase dicha validación no es correcta.

2.2 Requisitos no funcionales

Nombre	RNF – 1: Sistema integrado con EndaliaHR
Descripción	El sistema se integrará en EndaliaHR como un módulo más de la aplicación y podrá invocar a los otros módulos de EndaliaHR.

Nombre	RNF – 2: Sistema mantenible por el equipo de Endalia
Descripción	El sistema desarrollado será mantenible por el equipo de desarrolladores de Endalia. En la medida de lo posible se aplicarán tecnologías y herramientas conocidas.

Nombre	RNF – 3: Sistema intuitivo
Descripción	El sistema deberá ser fácil e intuitivo porque el rol de participante de un workflow abarca entre otros, perfiles no técnicos o usuarios no familiarizados con las tecnologías informáticas.

Nombre	RNF – 4: GUI web con los usuarios
Descripción	La interfaz de conexión con los usuarios se hará a través de web.

Nombre	RNF – 5: GUI coherente con EndaliaHR
Descripción	En la medida de lo posible, el estilo del GUI será similar y homogéneo con el utilizado en EndaliaHR.

Nombre	RNF – 6: Sistema escalable
Descripción	El sistema estará preparado para ampliar sus funcionalidades en un futuro.

Nombre	RNF – 7: Control de incidencias
Descripción	El sistema debe controlar las incidencias imprevistas que puedan surgir durante su ejecución y notificarlas a Endalia aportando información útil que pueda facilitar la detección y solución del error.

3. BIBLIOGRAFÍA

[1] IEEE, Recommended Practice fro Software Requirements Specifications, 1998.

endalia®
HR Consulting & Software

V. ESTÁNDAR DE DOCUMENTACIÓN

WORKFLOW MANAGEMENT SYSTEM
DAVID ROY

TABLA DE CONTENIDOS

Tabla de contenidos 2
1. Introducción 3
1.1 Objetivo 3
1.2 Alcance 3
1.3 Organización del documento 3
2. Formato de documentación 3
2.1 Fuentes y estilos 3
2.2 Interlineado y formato de párrafos 3
2.3 Imágenes y diagramas 4
3. Plantillas de documentación 5
3.1 Página 1 - Portada 5
3.2 Página 2 – Tabla de contenidos 6
3.3 Página 3 y siguientes – Página de contenido 7
4. Bibliografía 7

1. INTRODUCCIÓN

1.1 Objetivo

El objetivo de este documento es definir el estándar de documentación de este proyecto. En él se definen los formatos, diseños, tipología de fuentes y plantillas a utilizar en la documentación del proyecto.

1.2 Alcance

Las especificaciones de este documento deben aplicarse a toda la documentación generada durante el proyecto.

1.3 Organización del documento

El presente documento contiene las siguientes secciones:

1. Introducción: Descripción del objetivo y alcance del documento.
2. Formato de documentación: especificaciones sobre las fuentes y estilos que deben utilizarse.
3. Plantillas de documentación: detalle de las páginas que componen un fichero de documentación.
4. Bibliografía: Referencias a libros y webs utilizadas en el documento.

2. FORMATO DE DOCUMENTACIÓN

2.1 Fuentes y estilos

A continuación, se definen los tipos de fuente utilizados para los diferentes formatos de texto del documento:

- Título 1: *Helvética Neue* 16.
- Título 2: *Helvética Neue* 14.
- Titulo 3: *Helvética Neue* 12.
- Texto normal: *Helvética Neue* 10.
- Texto en pie de imágenes: *Helvética Neue* 9, Cursiva.
- Texto de código fuente o acciones de línea de comandos: *Courier New* 10.

2.2 Interlineado y formato de párrafos

El interlineado utilizado en la documentación será sencillo. El texto se justificará por ambos márgenes.

Se evitará dejar títulos de segundo y tercer nivel como última línea de una página, siendo ubicados en la página siguiente.

La separación de todos los títulos, independientemente del nivel, y el elemento que le precede será de dos saltos de línea.

En las listas de elementos cada ítem se indicará con un punto al principio de la línea. En el caso de listas anidadas, se indentarán mediante tabulados de la siguiente manera:

- Lista de nivel 1
    - Lista de nivel 2
        - Lista de nivel 3

2.3 Imágenes y diagramas

Las imágenes y diagramas se colocarán centrados y ajustando en la medida de lo posible su tamaño a los márgenes de la página, excepto en el caso de que su tamaño sea excesivo para visualizarlos de manera adecuada dentro de esos márgenes, en cuyo caso se colocarán en una página nueva en disposición horizontal. Todas las imágenes estarán numeradas y se les referenciará en el texto al que acompañan mediante una definición centrada debajo de las mismas, de la siguiente manera:

Figura 1: Ejemplo de imagen

3. PLANTILLAS DE DOCUMENTACIÓN

Las plantillas explicadas en esta sección están disponibles para su uso en formato electrónico.

El objetivo de este apartado es especificar el formato que deben cumplir los documentos que acompañen a este proyecto. Para cada plantilla se muestra una captura a tamaño reducido y una explicación de las partes que las componen y la manera de utilizarlas. Como ejemplo del aspecto final de dichas plantillas sirve el presente documento.

3.1 Página 1 - Portada

El formato de la portada de todos los documentos es el siguiente:

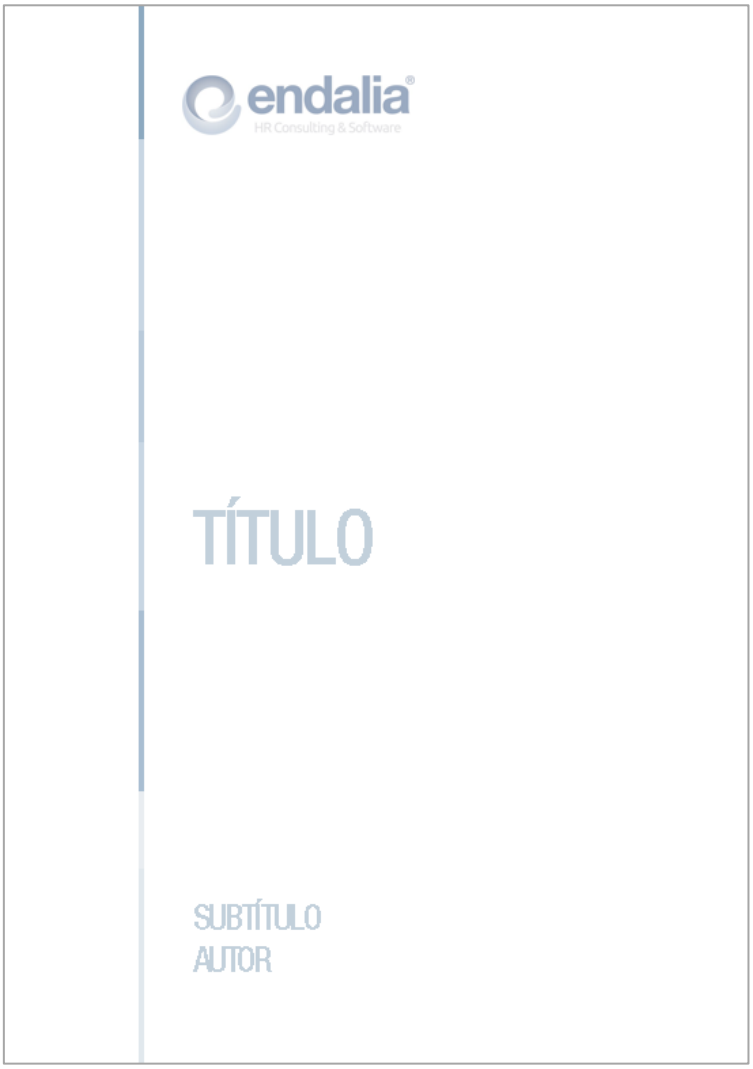


Figura 2: Ejemplo de portada

En la portada se indica el título del documento con fuente *Helvética Neue* 60 y el subtítulo y autor con *Helvética Neue* 28.

La portada no comparte cabecera ni pie de página con el resto de páginas del documento.

3.2 Página 2 – Tabla de contenidos

La página 2 muestra la tabla de contenidos del documento con el siguiente formato:

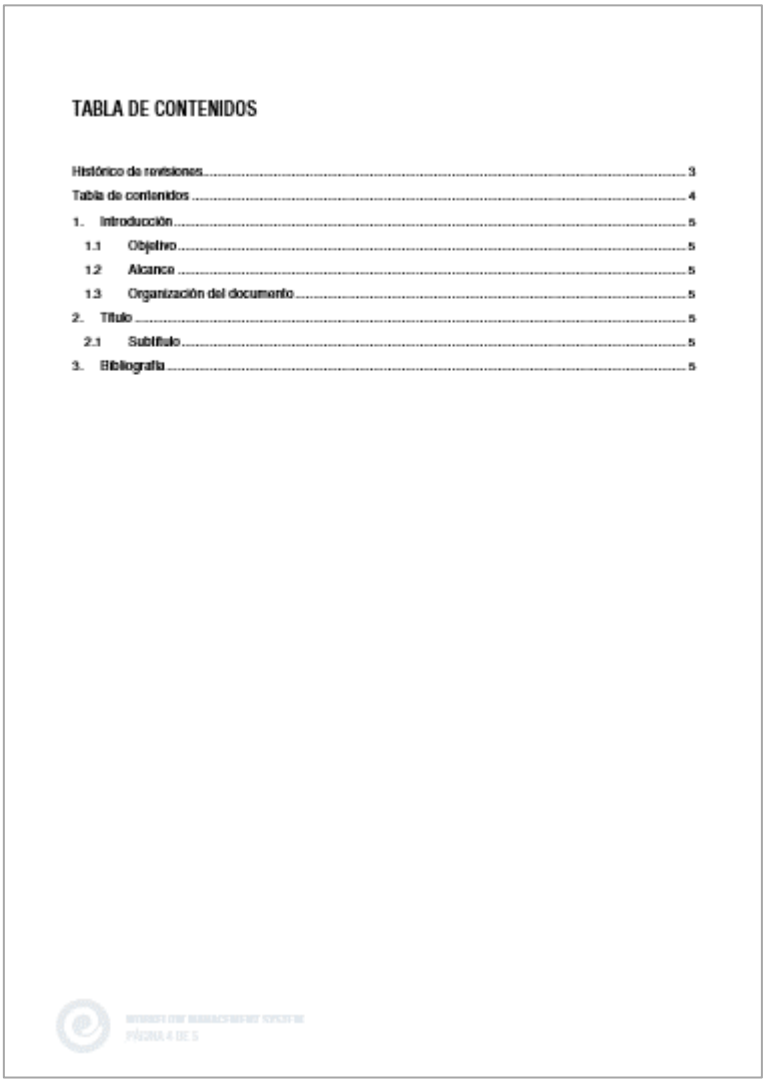


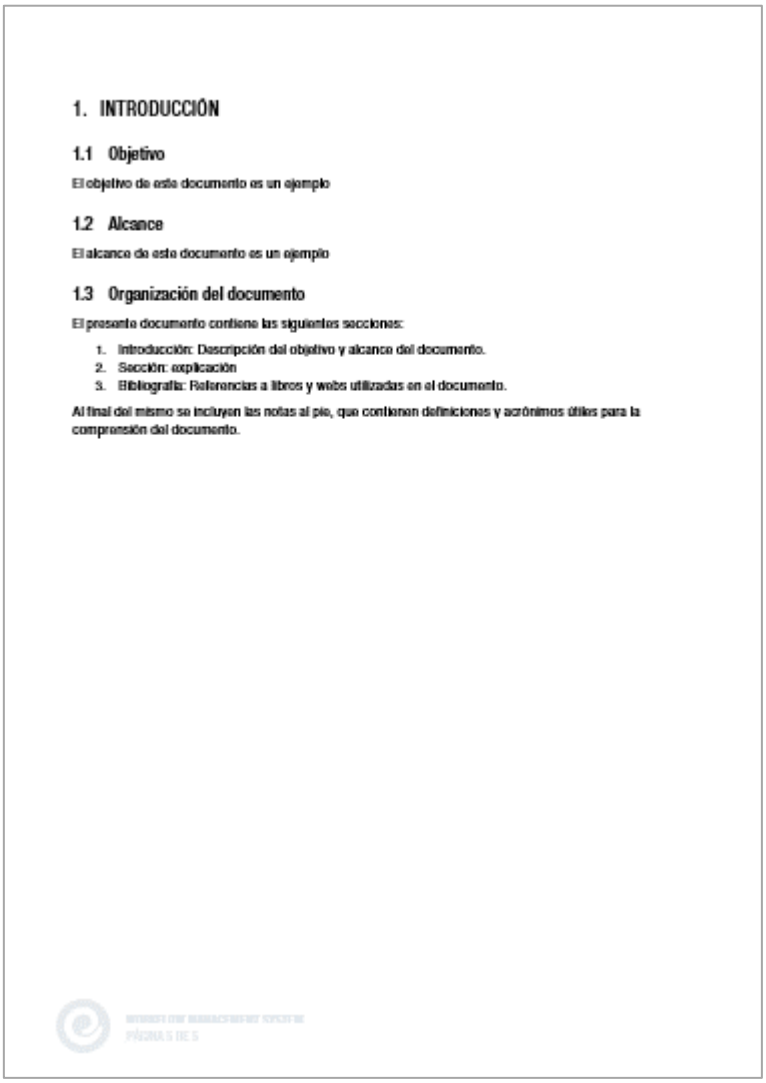
Figura 3: Tabla de contenidos

La tabla de contenidos se genera automáticamente con la herramienta Microsoft Word. El tipo de letra para los apartados de nivel 1 es *Helvética Neue* 10.

El pie de página es común para todas las hojas del documento a partir de la página 2, incluida. En él se muestra el logotipo de Endalia seguido del título del documento y la numeración de la página sobre el total de páginas que componen el documento.

3.3 Página 3 y siguientes – Página de contenido

A partir de la página 3 incluida, se añadirán las secciones propias del documento siguiendo las indicaciones descritas anteriormente. A continuación, se muestra un ejemplo ilustrativo:



1. INTRODUCCIÓN

1.1 Objetivo

El objetivo de este documento es un ejemplo

1.2 Alcance

El alcance de este documento es un ejemplo

1.3 Organización del documento

El presente documento contiene las siguientes secciones:

1. Introducción: Descripción del objetivo y alcance del documento.
2. Sección: explicación
3. Bibliografía: Referencias a libros y webs utilizadas en el documento.

Al final del mismo se incluyen las notas al pie, que contienen definiciones y acrónimos útiles para la comprensión del documento.

Figura 4: Página de contenido

4. BIBLIOGRAFÍA

[1] Edward J. Huth, Scientific Style and Format: The CBE Manual for Authors, Editors and Publishers, Cambridge University Press, 1994.

endalia®
HR Consulting & Software

VI. ESTÁNDAR DE CODIFICACIÓN

WORKFLOW MANAGEMENT SYSTEM

DAVID ROY

TABLA DE CONTENIDOS

Tabla de contenidos 2
1. Introducción 4
1.1 Objetivo 4
1.2 Alcance 4
1.3 Organización del documento 4
2. Estructura de los ficheros 4
2.1 Consideraciones generales 4
2.2 Codificación de archivos de código fuente 5
2.2.1 Sección using 5
2.2.2 Sección namespace y class 5
2.2.3 Región declaración de variables 6
2.3 Codificación de archivos de clase de acceso a datos 6
2.3.1 Sección using 7
2.3.2 Sección namespace y class 7
2.3.3 Región atributos 7
2.3.4 Región constructores 7
2.3.5 Región propiedades 7
2.3.6 Región métodos 8
2.4 Nomenclatura de recursos de internacionalización 8
3. Reglas de codificación 8
3.1 Indentación 8
3.1.1 Longitud de línea 8
3.1.2 Ruptura de líneas 9
3.2 Comentarios 9
3.2.1 Aplicación de los comentarios 9
3.2.2 Formatos de implementación de comentarios 10
3.3 Declaraciones 11
3.3.1 Número de declaraciones por línea 11
3.3.2 Inicialización 12
3.3.3 Situación 12
3.4 Sentencias 12
3.4.1 Sentencias simples 12
3.4.2 Sentencias compuestas 12
3.4.3 Sentencias de retorno 13

3.4.4 Sentencias if, if-else, if else- if else 13
3.4.5 Sentencias for 13
3.4.6 Sentencias while 14
3.4.7 Sentencias do-while 14
3.4.8 Sentencias switch 14
3.4.9 Sentencias try-catch 14
3.5 Espacios en blanco 15
3.5.1 Líneas en blanco 15
3.5.2 Espacios en blanco 15
3.6 Convenciones de nombres 16
3.6.1 Clases 16
3.6.2 Métodos 16
3.6.3 Variables y parámetros 16
3.6.4 Constantes 17
3.7 Hábitos de programación 17
3.7.1 Referencias a variables y métodos de clase 17
3.7.2 Constantes 17
3.7.3 Asignaciones de variables 17
3.7.4 Paréntesis 18
3.7.5 Variables de retorno 18
3.7.6 Operador condicional 18
3.7.7 Comentarios especiales 18
4. Estándar de nombrado de Base de Datos 19
4.1 Idioma a utilizar 19
4.2 Convenciones de nombrado de tablas 19
4.2.1 Nombrado de tablas de entidad 19
4.2.2 Nombrado de tablas de relación 19
4.2.3 Nombres de tablas predefinidos 19
4.3 Convenciones de nombrado de campos 20
4.3.1 Nombrado de Campos de Tablas de Entidad 20
4.3.2 Nombrado de campos de tablas de relación 20
5. Bibliografía 21

1. INTRODUCCIÓN

1.1 Objetivo

El presente documento describe las normas que deben seguirse en el desarrollo de cualquier tipo de elemento de codificación realizado en este proyecto. Estas normas y convenciones permiten homogeneizar el proceso de implementación y lograr con ello los siguientes objetivos:

- Facilitar la verificación y validación del código.
- Mejorar el mantenimiento, gracias a que el código es fácilmente legible por cualquier desarrollador.
- Aplicar un formato adecuado para la distribución del código fuente como producto.

1.2 Alcance

Este documento es un elemento necesario previo a la realización de cualquier tipo de código fuente del proyecto, y será utilizado como guía durante toda la fase de construcción o implementación.

1.3 Organización del documento

El presente documento contiene las siguientes secciones:

1. Introducción: Descripción del objetivo y alcance del documento.
2. Estructura de los ficheros: describe las distintas secciones en que se dividen los diferentes tipos de ficheros de código fuente.
3. Reglas de codificación: normas y formatos que deben aplicarse a los distintos componentes del código fuente, como son las instrucciones, los comentarios, las declaraciones, etc.
4. Estándar de nombrado de Base de Datos: convenciones para designar los nombres de las tablas y campos de base de datos.
5. Bibliografía: Referencias a libros y webs utilizadas en el documento.

2. ESTRUCTURA DE LOS FICHEROS

En este apartado se define la estructura y organización de los archivos de código fuente del proyecto. Asimismo, se especifica la codificación para los archivos de clases y acceso a datos y la nomenclatura de los recursos de internacionalización.

2.1 Consideraciones generales

El lenguaje C# proporciona las directivas `#region` y `#endregion` que permiten agrupar varias líneas contiguas de código en una sección que puede identificarse con un nombre. Con ello, el editor de código Visual Studio permite visualmente compactar y expandir las regiones, ayudando a la legibilidad del código.

En los apartados siguientes se especifican las distintas regiones que deben tener los diferentes tipos de ficheros de código. Esas regiones son obligatorias en el caso de que aparezcan los elementos para los que

han sido definidas. En caso de que aparezcan elementos no especificados en ninguna de las regiones, podrán definirse nuevas regiones para dichos elementos. En cualquier caso, no se permitirán métodos o funciones que no estén incluidos dentro de alguna región.

2.2 Codificación de archivos de código fuente

A continuación, se muestra un ejemplo de código fuente con la estructura estándar. En la parte de la izquierda se muestra el nombre de cada sección del código. Los detalles de codificación de cada sección se muestran en los siguientes apartados de este documento:

Sección	Código fuente
Directivas *using*	`using System;` `<directivas using>`

Declaración *namespace* y *class*	`namespace MyNamespace1`
	`{`
	`  public class MyClass`
	`  {`

Región variables	`  #region variables`
	`  #region Variables I18N`
	`  <declaraciones de variables>`
	`  #endregion Variables I18N`
	`  #region Variables globales`
	`  <declaraciones de variables>`
	`  #endregion Variables globales`
	`  #endregion variables`

Región *i18n*[1]	`  #region I18N`
	`  <Código InitializeLabels>`
	`  <Código LoadI18N>`
	`  #endregion I18N`

	`  }` `}`

2.2.1 Sección *using*

En esta sección se colocarán, por orden alfabético creciente, las directivas que especifiquen las clases utilizadas en el código fuente.

2.2.2 Sección *namespace* y *class*

En esta sección se colocarán las cabeceras que especifican el espacio de nombres al que pertenece el código y el nombre de la clase. Esta última irá precedida por una cabecera en la que se especificarán los siguientes datos:

[1] i18n: *Internationalization*. Es una práctica común en el idioma inglés (sobre todo en el ámbito de la computación), abreviar *internationalization* con el numerónimo "i18n". Ello se debe a que entre la primera i y la última ene de dicha palabra hay 18 letras.

```
/// <summary>
/// Nombre del fichero : Nombre del fichero
/// Autor              : Nombre del autor
/// Descripción        : Descripción de la funcionalidad y objetivo del fichero
/// Copyright © 2017, Endalia, S.L. Todos los derechos reservados
/// </summary>
```

2.2.3 Región declaración de variables

Esta región estará integrada por 4 subregiones que se especifican a continuación:

- Variables i*18n*: en esta región aparecen las declaraciones de cadenas que se utilizan para definir todos los textos que son presentados al usuario.
- Variables globales: en esta región aparecen las declaraciones de variables globales acompañadas de una descripción de su funcionalidad.
- Región i*18n*: En esta región se colocan dos métodos:
    - Método *InitializeLabels*: Realiza la lectura del fichero de recursos que almacena las cadenas de internacionalización, cargando éstas en variables de cadena que llamaremos «*etiquetas*»**.**
    - Método *LoadI18N*: Carga en los campos de texto del formulario las *etiquetas*, las variables de cadenas obtenidas en el método *InitializeLabels*.

2.3 Codificación de archivos de clase de acceso a datos

A continuación, se muestra la estructura de codificación de los archivos de acceso a datos y definición de clase. En la columna izquierda se muestra el nombre de la sección correspondiente del código:

Sección	Código fuente
Directivas *using*	`using System;` `<directivas using>`

Declaración *namespace* y *class*	`namespace MyNamespace1`
	`{`
	`  public class MyClass`
	`  {`

Región atributos	`  #region atributos`
	`  <declaraciones de atributos>`
	`  #endregion atributos`

Región constructores	`  #region constructores`
	`  <Métodos constructores de la clase>`
	`  #endregion constructores`

Región propiedades	`  #region propiedades`
	`  <declaraciones de propiedades>`
	`  #endregion propiedades`

Región métodos	`  #region métodos`
	`  <Código métodos>`
	`  #endregion métodos`

	`  }` `}`

2.3.1 Sección *using*

En esta sección se colocarán, por orden alfabético creciente, las directivas que especifiquen las clases utilizadas en el código fuente.

2.3.2 Sección *namespace* y *class*

En esta sección se colocarán las cabeceras que especifican el espacio de nombres al que pertenece el código y el nombre de la clase. Esta última irá precedida por una cabecera en la que se especificarán los siguientes datos:

```
/// <summary>
/// Nombre del fichero : Nombre del fichero
/// Autor              : Nombre del autor
/// Descripción        : Descripción de la funcionalidad y objetivo del fichero
/// Copyright © 2017, Endalia, S.L. Todos los derechos reservados
/// </summary>
```

2.3.3 Región atributos

En esta región se colocará la declaración de los atributos de una clase. Los atributos se nombrarán mediante Camel-Casing[2] y precedidos por un guión bajo de este modo:

```
private int _userID;
```

Asimismo, en esta región se declararán las constantes de la clase, que se nombrarán con mayúsculas.

2.3.4 Región constructores

En esta región se colocará la declaración de los métodos constructores de la clase.

2.3.5 Región propiedades

Las propiedades se nombran con Pascal-Casing[3] y, en el caso de que representen el acceso al valor de un atributo, su nombre es el mismo del atributo sin el guión bajo inicial y con la primera letra en mayúscula:

```
public int UserID
{
	get{ return _userID; }
	set{ _userID = value; }
}
```

---

[2] Camel-Casing: Notación en la que un identificador está compuesto por múltiples palabras juntas, sin espacios ni guiones entre ellas, comenzando cada una de ellas por mayúscula a excepción de la letra inicial.

[3] Pascal-Casing: Notación en la que un identificador está compuesto por múltiples palabras juntas, sin espacios ni guiones entre ellas, comenzando cada una de ellas por mayúscula.

2.3.6 Región métodos

En esta región se colocará la declaración de los métodos de la clase.

2.4 Nomenclatura de recursos de internacionalización

Como se ha indicado en los apartados anteriores, los recursos de internacionalización se cargan en variables de cadena que llamaremos genéricamente *«etiquetas»*. El nombrado de estas etiquetas se hace del siguiente modo:

- La parte inicial del nombre de la etiqueta será la misma que el nombre del archivo fuente en el que se utilizará la etiqueta. A continuación, se colocará un guión bajo seguido de un prefijo que indicará la utilización de la etiqueta seguida de un guión bajo, siguiendo la siguiente convención:
    - etiqueta o campo de texto: `_lbl_`
    - etiqueta de hyperlink: `_lnk_`
    - texto de botón: `_btn_`
- En el caso de que la etiqueta sea un *tooltip*[4], se colocará a continuación el prefijo `_toolTip_`
- Por último, se colocará un nombre descriptivo de la función de la etiqueta en notación Pascal-Casing.

Un ejemplo de nombre de etiqueta es:

```
UsersList_btn_toolTip_AddUser
```

3. REGLAS DE CODIFICACIÓN

3.1 Indentación

Dada la actual uniformidad y estandarización de los editores utilizados para el desarrollo de código C# en .NET, se utilizará el tabulador como unidad de indentación estándar. Los comentarios se indentarán al mismo nivel de indentación que el código que se esté documentando.

3.1.1 Longitud de línea

Se recomienda no escribir líneas con más de 80 caracteres, ya que no son bien manejadas por muchos terminales y herramientas.

[4] Tooltip: Elemento de ayuda visual, que funciona al situar el cursor sobre algún elemento gráfico, mostrando un contenido adicional para informar al usuario de la finalidad del elemento sobre el que se encuentra.

3.1.2 Ruptura de líneas

Cuando una expresión no quepa en una sola línea, se debe fraccionar o *romper* de acuerdo a estos principios generales:

- Romper después de una coma.
- Romper antes de un operador.
- Preferir las rupturas de alto nivel a las de bajo nivel.

Alinear la nueva línea con el principio de la expresión al mismo nivel de la línea anterior.

3.2 Comentarios

En C# hay tres formas de escribir comentarios:

- Bloque: encerrar todo el texto que se desee comentar entre los caracteres /* y */:

  /*<texto>*/

  Estos comentarios pueden abarcar tantas líneas como sea necesario.

  No es posible anidar comentarios de este tipo.

- Línea: añadir los caracteres // justo antes del texto que se desea comentar:

  // <texto>

  El comentario abarcará desde los caracteres // hasta el final de la línea.

- Documentación: la tercera manera de escribir un comentario es utilizando el trío de caracteres ///.

  /// <texto>

  Aplican las mismas reglas que para los comentarios de línea, pero tiene la particularidad de ser reconocido y utilizado por las herramientas de generación automática de documentación y será el utilizado para la descripción de métodos y clases, ya que en el caso de los primeros genera la estructura de etiquetas o *tags* de documentación de nombres, parámetros y valores de retorno utilizados para la documentación automatizada.

3.2.1 Aplicación de los comentarios

Aparte del uso ya descrito de los comentarios de documentación (///), los comentarios deberían usarse para una introducción del código y proporcionar información adicional que no está disponible en el propio código. Los comentarios sólo deberían tener información que sea relevante para leer y entender el programa. Por ejemplo, información sobre cómo está construida la clase correspondiente o en qué directorio reside no debería ser incluida como comentarios.

Las discusiones no triviales o decisiones de diseño no obvias son apropiadas, pero debemos evitar la duplicidad de información que esté presente en el código. Es demasiado fácil que los comentarios redundantes se queden anticuados. En general, debemos evitar cualquier comentario que se pueda quedar anticuado cuando el código evolucione.

La frecuencia en los comentarios algunas veces refleja una pobre calidad de código. Cuando nos sintamos obligados a llenarlo de comentarios, debemos considerar la reescritura del código para hacerlo más claro. Los

comentarios no deben encerrarse en grandes cajas dibujadas con asteriscos u otros caracteres. Los comentarios nunca deberían incluir caracteres especiales como saltos de página, etc.

Los siguientes puntos son técnicas de comentarios recomendadas.

- Cuando se modifica el código, se mantienen siempre actualizados los comentarios circundantes.
- Evitar los comentarios recargados, como las líneas enteras de asteriscos. En su lugar se utilizan espacios para separar los comentarios y el código.
- Evitar rodear un bloque de comentarios con un marco tipográfico. Puede resultar agradable, pero es difícil de mantener.
- Antes de la implementación, quitar todos los comentarios temporales o innecesarios, para evitar cualquier confusión en la futura fase de mantenimiento.
- Si se necesita realizar comentarios para explicar una sección de código compleja, examinar el código para decidir si se debería volver a escribir. Siempre que sea posible, no documentar un código malo, sino volver a escribirlo. Aunque, por regla general, no debe sacrificarse el rendimiento para hacer un código más simple para el usuario, es indispensable un equilibrio entre rendimiento y mantenibilidad.
- Usar frases completas en los comentarios y evitar ambigüedades.
- Ir comentando al mismo tiempo que se programa, porque probablemente no habrá tiempo de hacerlo más tarde. Por otro lado, aunque se tuviera oportunidad de revisar el código que se ha escrito, lo que parece obvio hoy es posible que seis semanas después no lo sea.
- Evitar comentarios superfluos o inapropiados, como comentarios divertidos al margen.
- Usar los comentarios para explicar el propósito del código como si fueran traducciones interlineales.
- Comentar cualquier cosa que no sea legible de forma obvia en el código.
- Para evitar problemas recurrentes, hacer siempre comentarios al depurar errores y solucionar problemas de codificación, especialmente cuando se trabaje en equipo.
- Hacer comentarios en el código que esté formado por bucles o bifurcaciones lógicas. Se trata en estos casos de áreas clave que ayudarán a los lectores del código fuente.
- Realizar los comentarios en un estilo uniforme, respetando una puntuación y estructura coherentes a lo largo de toda la aplicación.
- Separar los comentarios de sus delimitadores mediante espacios. Si se respeta esta norma, los comentarios serán más claros y fáciles de localizar si trabaja sin indicaciones de color.

3.2.2 Formatos de implementación de comentarios

Los programas pueden tener cuatro estilos de implementación de comentarios:

- Documentación: Los comentarios iniciados con /// antes de la declaración de un método, usando Visual Studio, generan de forma automática el siguiente bloque con la información del método, parámetros y valor devuelto. Como ya se indicó anteriormente, estos comentarios son utilizados por los generadores automáticos de documentación.

```
/// <summary>
/// Deletes a user
/// </summary>
/// <param name="userID">The id of the user</param>
/// <returns>False if user not found or not deleted.</returns>
public bool DeleteUser(int userID)
```

- Bloque de comentarios: Los bloques de comentarios se usan para proporcionar descripciones de ficheros, estructuras de datos y algoritmos. Los bloques de comentarios podrían usarse al principio de cada fichero y antes de cada declaración de clase. También pueden usarse en otros lugares, como dentro de los métodos. Para este tipo de comentario se preferirá la estructura /* - */. Un bloque de comentario debería ir precedido por una línea en blanco para separarlo del resto del código:

```
/*
 * Esto es un bloque de comentarios.
 */
```

- Comentarios de una línea: Los comentarios cortos pueden aparecer como una sola línea indentada al nivel del código que la sigue. Si un comentario no se puede escribir en una sola línea, debería seguir el formato de los bloques de comentario. Un comentario de una sola línea debería ir precedido de una sola línea en blanco. Para este tipo de comentario se preferirá utilizar los caracteres '//'. A continuación, se muestra un ejemplo:

```
if (condition)
{
      // Código de la condición.
      ...
}
```

- Comentarios finales: Los comentarios muy cortos pueden aparecer en la misma línea que el código que describen, pero deberían separarse lo suficiente de las sentencias. Si aparece más de un comentario en el mismo trozo de código, deberían estar indentados a la misma altura. Para este tipo de comentario se preferirá utilizar los caracteres '//'. Aquí tenemos un ejemplo utilizando estos caracteres y la estructura /* - */:

```
if (a == 2)
{
        return TRUE;                    // caso especial
}
else
{
         return isPrime(a);             /* otro comentario */
}
```

3.3 Declaraciones

3.3.1 Número de declaraciones por línea

Se recomienda una declaración por línea porque mejora la legibilidad y permite añadir comentarios individuales a cada declaración. Ejemplo:

```
int level;       // nivel de identación
int size;        // tamaño
```

es preferible antes que:

```
int level, size;
```

En cualquier caso, nunca debemos poner diferentes tipos en la misma línea. Por ejemplo:

```
int foo, fooarray[]; //Evitar
```

3.3.2 Inicialización

Debemos intentar inicializar las variables locales donde son declaradas. La única razón para no inicializar una variable donde es declarada es si el valor inicial depende de algún cálculo que tiene que ocurrir antes.

3.3.3 Situación

Ponemos las declaraciones sólo al principio de los bloques. No debemos esperar a declarar variables hasta que son usadas por primera vez; puede confundir al programador y estorbar la portabilidad del código dentro del ámbito.

```
void myMethod()
{
  int int1 = 0;           // comienzo de bloque

  if (condition)
  {
        int int2 = 0;     // comienzo de bloque if
        ...
  }
}
```

La única excepción a esta regla son los indexados para los bucles, que en C# pueden ser declarados en la sentencia for:

```
for (int i = 0; i < maxLoops; i++) { ... }
```

Debemos evitar las declaraciones locales que oculten las declaraciones de nivel superior. Por ejemplo, no debemos declarar el mismo nombre de variable en un bloque interno:

```
int count;

myMethod() {

  if (condition) {

    int count;     // Evitar

  }

}
```

3.4 Sentencias

3.4.1 Sentencias simples

Cada línea debe contener, como máximo, una sentencia. Por ejemplo:

```
argv++;          // Correcto
argc++;          // Correcto
argv++; argc--;  // Evitar
```

3.4.2 Sentencias compuestas

Las sentencias compuestas son sentencias que contienen listas de sentencias encerradas entre llaves '{ …}'.

- Las sentencias encerradas deben indentarse uno o más niveles que la sentencia compuesta.

- La llave de apertura '{' debe empezar una nueva línea a continuación de la que empieza la sentencia compuesta; la llave de cierre '}' debe empezar una nueva línea y estar indentada con el principio de la sentencia compuesta.
- Las llaves se usan alrededor de todas las sentencias, incluso para sentencias simples, cuando éstas forman parte de una estructura de control como una sentencia *if-else* o *for*. Esto hace más fácil la adición de sentencias sin introducir errores debido al olvido de las llaves.

Las únicas excepciones a esta regla serán para los métodos *get* y *set* de las clases de acceso a datos descritos en el apartado 2.3.5

3.4.3 Sentencias de retorno

Una sentencia de retorno no deberá usar paréntesis a menos que el valor de retorno sea más obvio de esta forma. Por ejemplo:

```
return;
return myDisk.size();
return (size ? size : defaultSize);
```

3.4.4 Sentencias *if, if-else, if else- if else*

Las sentencias de tipo *if-else* deben tener la siguiente forma:

```
if (condition)
{
  statements;
}
else if (condition)
{
  statements;
}
else
{
  statements;
}
```

Las sentencias *if* siempre usan llaves. Debemos evitar el siguiente caso:

```
if (condition) //Evitar, se han omitido las llaves {}!
  statement;
```

Aunque es perfectamente válido a nivel de código, puede llevar a confusión y a la introducción de errores.

3.4.5 Sentencias *for*

Una sentencia *for* debe tener la siguiente forma:

```
for (initialization; condition; update)
{
  statements;
}
```

Una sentencia *for* vacía, en la cual todo el trabajo se hace en las cláusulas de inicialización, condición y actualización, debe tener la siguiente forma:

```
for (initialization; condition; update);
```

Cuando usamos el operador de asignación '=' en las cláusulas de inicialización o actualización de un *for*, debemos evitar usar más de tres variables. Si es necesario, debemos usar sentencias separadas antes del bucle *for*, para la cláusula de inicialización, o al final del bucle, para la cláusula de actualización.

3.4.6 Sentencias *while*

Una sentencia *while* debe tener la siguiente forma:

```
while (condition)
{
    statements;
}
```

Una sentencia *while* vacía debe tener la siguiente forma:

```
while (condition);
```

3.4.7 Sentencias *do-while*

Una sentencia *do-while* debe tener la siguiente forma:

```
do
{
  statements;
}
while (condition);
```

3.4.8 Sentencias *switch*

Una sentencia *switch* debe tener la siguiente forma:

```
switch (expression)
{
      case constant-expression:
            statement
            break;

      case constant-expression:
            statement
            /* continúa sin break */

       [default:
            statement
            jump-statement]
}
```

Cada vez que un *case* no incluye una sentencia *break*, debemos añadir un comentario donde normalmente iría la sentencia *break*. En el ejemplo anterior se puede observar este comportamiento. En cualquier caso, se deberá evitar este tipo de construcción.

Toda sentencia *switch* debe incluir un valor *default*. El *break* en este caso podría parecer redundante, pero podría evitar un error si añadimos después otro *case*.

3.4.9 Sentencias *try-catch*

Una sentencia *try-catch* debe tener la siguiente forma:

```
try
{
  statements;
```

```
}
catch (Exception e)
{
  statements;
}
```

Una sentencia *try-catch* también puede ir seguida de un bloque *finally*, que se ejecuta sin importar si se ha completado con éxito o no el bloque try.

```
try
{
  statements;
}
catch (Exception e)
{
  statements;
}
finally
{
  statements;
}
```

3.5 Espacios en blanco

3.5.1 Líneas en blanco

Las líneas en blanco mejoran la lectura separando secciones de código que están relacionadas lógicamente.

Siempre se debe usar dos líneas en blanco en las siguientes circunstancias:

- Entre secciones de un fichero fuente.
- Entre definiciones de clases e interfaces.

Siempre se debe usar una línea en blanco en las siguientes circunstancias:

- Entre métodos.
- Entre las variables locales de un método y su primera sentencia.
- Antes de un bloque de comentarios o un comentario simple.
- Entre secciones lógicas dentro de un método para mejorar su lectura.

3.5.2 Espacios en blanco

Los espacios en blanco deben usarse en las siguientes circunstancias:

- Una palabra clave seguida por un paréntesis deben estar separados por un espacio en blanco:

```
while (true)
{
...
}
```

- No se debe usar un espacio en blanco entre un nombre de método y su paréntesis de apertura. Esto ayuda a distinguir las palabras clave de las llamadas a métodos.
- Después de las comas en una lista de argumentos debe aparecer un espacio en blanco.
- Todos los operadores binarios excepto “.” Deben estar separados de sus operandos por espacios. Los espacios en blanco nunca deben separar los operadores unarios como incremento “++” y

decremento “—“ de sus operadores. Por ejemplo:

```
a += c + d;
a = (a + b) / (c * d);
while (d++ = s++)
{
    n++;
}
```

- Las expresiones de una sentencia deben estar separadas por espacio. Por ejemplo:

```
for (expr1; expr2; expr3)
```

3.6 Convenciones de nombres

3.6.1 Clases

Se debe usar Pascal-Casing para el nombrado de clases. Debemos intentar mantener los nombres de clases simples y descriptivos. Debemos usar palabras completas y evitar acrónimos y abreviaturas (a menos que la abreviatura se use muy ampliamente como URL o HTML). Ejemplo:

```
public class HelloWorld
{
       ...
}
```

3.6.2 Métodos

Los métodos deben ser verbos. Se debe usar Pascal-Casing para su nombrado. Ejemplo:

```
public class HelloWorld
{
       void SayHello(string name)
       {
              ...
       }
}
```

3.6.3 Variables y parámetros

Se usa Camel-Casing para el nombrado de variables y parámetros. Los nombres de variables deben ser cortos y llenos de significado. La elección de una variable debe ser mnemónica, es decir, diseñada para indicar al observador casual su utilización. Se deben evitar los nombres de variable de un sólo carácter, excepto para variables temporales. Algunos nombres comunes de este tipo de variables son: i, j, k, m, y n para enteros. Ejemplo:

```
public class HelloWorld
{
       int totalCount = 0;
       void SayHello(string name)
       {
              string fullMessage = "Hello " + name;
       }
}
```

3.6.4 Constantes

Los nombres de constantes deben escribirse en mayúsculas con las palabras separadas por guiones bajos "_". Ejemplo:

```
public static int MIN_WIDTH = 4;
public static int MAX_WIDTH = 999;
```

3.7 Hábitos de programación

No debemos hacer pública ninguna variable de instancia o de clase sin una buena razón. A menudo las variables de instancia no necesitan ser asignadas ni consultadas explícitamente. Normalmente, esto sucede como efecto lateral de llamadas a métodos. Un ejemplo apropiado de una variable de instancia pública es el caso en que la clase es esencialmente una estructura de datos, sin comportamiento.

3.7.1 Referencias a variables y métodos de clase

Evitar usar un objeto para acceder a una variable o método de tipo *static*. Usar el nombre de la clase en su lugar. Por ejemplo:

```
classMethod();            //correcto
AClass.classMethod();     //correcto
anObject.classMethod();   //Evitar
```

3.7.2 Constantes

Los valores constantes numéricos no se deben codificar directamente, en su lugar se deben usar constantes con dicho valor numérico asignado. La excepción a esta regla son los valores -1, 0 y 1, que pueden aparecer en un bucle *for* como contadores.

3.7.3 Asignaciones de variables

Evitar asignar el mismo valor a varias variables en la misma sentencia porque complica la lectura del código. Ejemplo:

```
fooBar.fChar = barFoo.lchar = 'c'; // Evitar
```

No usar el operador de asignación en un lugar donde se pueda confundir con el de igualdad. Ejemplo:

```
if (c++ = d++) { // Evitar
  ...
}
```

se debe escribir:

```
if ((c++ = d++)!= 0)
{
  ...
}
```

No debemos usar asignaciones embebidas como un intento de mejorar el rendimiento en tiempo de ejecución. Ese es el trabajo del compilador. Ejemplo:

```
d = (a = b + c) + r; // Evitar
```

Debería escribirse como:

```
a = b + c;
d = a + r;
```

3.7.4 Paréntesis

En general es una buena idea usar paréntesis en expresiones que implican distintos operadores para evitar problemas con el orden de precedencia de los operadores. Incluso si parece claro el orden de precedencia de los operadores, podría no ser así para otros. No se debe asumir que otros programadores conozcan el orden de precedencia.

```
if (a == b && c == d)      // Evitar
if ((a == b) && (c == d)) // Correcto
```

3.7.5 Variables de retorno

Debemos intentar hacer que la estructura de nuestro programa se corresponda con nuestra intención. Por ejemplo:

```
if (booleanExpression)
{
       return true;
}
else
{
       return false;
}
```

debería escribirse:

```
return booleanExpression;
```

De forma similar,

```
if (condition)
{
       return x;
}
return y;
```

debería escribirse como:

```
  return (condition ? x : y);
```

3.7.6 Operador condicional

Si una expresión contiene un operador binario antes de `?` en el operador ternario `?:`, se debe colocar entre paréntesis. Ejemplo:

```
(x >= 0) ? x : -x;
```

3.7.7 Comentarios especiales

Debemos usar la etiqueta `XXX` en un comentario para indicar que algo tiene algún error pero funciona. Por otro lado, debemos usar la etiqueta `FIXME` para marcar algo que tiene algún error y no funciona.

4. ESTÁNDAR DE NOMBRADO DE BASE DE DATOS

4.1 Idioma a utilizar

Para el nombrado de todos los elementos de la base de datos, el idioma a utilizar será el inglés**,** salvo que se especifique de manera explícita lo contrario**.**

4.2 Convenciones de nombrado de tablas

4.2.1 Nombrado de tablas de entidad

El nombrado de las tablas de entidad dentro de la base de datos seguirá la siguiente convención:

- Todos los nombres comenzarán por tres letras descriptivas del módulo o ámbito de aplicación de la tabla. La primera de estas tres letras será mayúscula y las otras dos restantes serán minúsculas. Los prefijos generales predefinidos, con una descripción de los mismos, se encuentran en el apartado 4.2.3 de este documento. En el caso de que el ámbito de aplicación de la tabla a nombrar difiera de los que aparecen en este apartado se definirá un nuevo prefijo y se añadirá a los existentes.
- En el caso de que la tabla forme parte del desarrollo de una sección en concreto dentro de un módulo, o forme parte de un conjunto de tablas relacionadas entre sí dentro de un ámbito de aplicación de la base de datos, se colocarán después del prefijo anterior y separadas por un guion bajo '_' tres letras descriptivas de la sección. La primera de estas letras será mayúscula y las otras dos restantes serán minúsculas. Si la tabla no tuviese relación con otras dentro del ámbito definido por el prefijo inicial estas tres letras no son necesarias.
- A continuación y separado por un guión bajo '_' se colocará un nombre descriptivo en plural del contenido de la tabla. Se deberá usar Pascal-Casing para este el nombrado. Se deben mantener los nombres simples y descriptivos. Se deben usar palabras completas y evitar acrónimos y abreviaturas (a menos que la abreviatura se use muy ampliamente como ID, URL o HTML, en cuyo caso puede escribirse en mayúsculas).

  Ejemplos:

```
Gen_SystemOptions
Orh_Emp_Employees
```

4.2.2 Nombrado de tablas de relación

Las tablas de relación comenzarán con un prefijo 'R_'.

A continuación, separados por guión bajo '_' y en singular, se colocarán utilizando Pascal-Casing los nombres descriptivos de las tablas que relaciona, pudiendo reducirse dichos nombres si alguno de ellos es muy largo.

De este modo, para relacionar las tablas 'Orh_Emp_Employees' y 'Orh_Job_Jobs' se creará la tabla 'R_Employee_Job'.

4.2.3 Nombres de tablas predefinidos

- R : Relación. Tablas de relación.

- Gen : Generic. Tablas relativas a la organización general de la base de datos.
- Orh : Organización y recursos humanos.
- Orh_Wfl_ : Tablas relativas a workflows.

4.3 Convenciones de nombrado de campos

Una norma establecida es que no puede haber dos campos dentro de una misma base de datos con el mismo nombre. Asimismo, el orden de los campos de una tabla debe seguir un orden lógico, (no tiene sentido colocar, <nombre, teléfono, apellidos, fax...>, sino <nombre, apellidos, teléfono, fax…>) Los nombres de los campos, dependiendo de si pertenecen a una tabla de entidad o de relación, se construyen siguiendo las normas de los siguientes apartados:

4.3.1 Nombrado de Campos de Tablas de Entidad

Si el campo es una clave, primaria o ajena, tiene unos determinados prefijos y sufijos:

- Claves primarias: Las claves primarias comienzan con el prefijo 'pk_' y terminan con el sufijo 'ID'.
- Claves ajenas: Las claves ajenas comienzan con el prefijo 'fk_' y si apunta a una clave primaria de otra tabla, entonces tiene el sufijo 'ID'.

A continuación del prefijo de clave, todos los campos de una misma tabla tienen un mismo prefijo que resume el nombre descriptivo de la tabla. La primera de estas letras es mayúscula y las restantes son minúsculas. Por ejemplo, si la tabla es 'Orh_Reg_DataBind' todos los campos de la tabla tendrán el prefijo 'Dbn' o 'Dbind'. Otro ejemplo: Si la tabla es 'Prj_Diary_Notes' los campos de la tabla tendrán el prefijo 'Diary' o 'Dia'.

Tras el prefijo de tabla se coloca el nombre del campo. Para nombrar los campos de la tabla se utilizan nombres descriptivos de su significado. Se debe usar Pascal-Casing para este nombrado. Se deben mantener los nombres simples y descriptivos. Se deben usar palabras completas y evitar acrónimos y abreviaturas (a menos que la abreviatura se use muy ampliamente como ID, URL o HTML, en cuyo caso puede escribirse en mayúsculas).

Ejemplos:

```
pk_EmpID
fk_EmpContractTypeID
EmpName
EmpSurname
```

4.3.2 Nombrado de campos de tablas de relación

En las tablas de relación, la nomenclatura de las claves ajenas y de los campos de datos sigue formatos diferentes:

- Las claves ajenas comienzan por 'fk_', a continuación los nombres descriptivos de la tablas que se relacionan (pudiendo resumirse si son muy largos), utilizando Pascal-Casing y sin separación entre ellos. A continuación el guión bajo de separación '_' seguido del nombre descriptivo de la tabla a la que apunta y se finaliza con el sufijo 'ID'.

  Ejemplo:

  Las claves ajenas de la tabla de relación 'R_EmployeeJob', que relaciona las tablas 'Orh_Emp_Employees' y 'Orh_Job_Jobs' son:

```
fk_EmpJobEmpID
fk_EmpJobJobID
```

- Los campos de datos en las tablas de relación se nombrarán mediante Pascal-Casing. El prefijo se forma con los nombres descriptivos de las tablas de entidad que se relacionan (pudiendo resumirse si son muy largos). Tras el prefijo de tabla se indica el nombre del campo. Este nombre debe ser descriptivo y simple. Se deben usar palabras completas y evitar acrónimos y abreviaturas (a menos que la abreviatura se use muy ampliamente como ID, URL o HTML, en cuyo caso puede escribirse en mayúsculas).

  Ejemplo:

  Algunos campos de datos de la tabla de relación 'R_EmployeeJob', que relaciona las tablas 'Orh_Emp_Employees' y 'Orh_Job_Jobs' son:

  ```
  EmpJobEndDate
  EmpJobIsTemporary
  EmpJobRemarks
  ```

5. BIBLIOGRAFÍA

[1] Juval Lowy, C# Coding Standard, http://www.sourceformat.com/pdf/cs-coding-standard-idesign.pdf: Idesign Inc, 2005.

endalia®
HR Consulting & Software

VII. ANÁLISIS Y DISEÑO

WORKFLOW MANAGEMENT SYSTEM

DAVID ROY

TABLA DE CONTENIDOS

Tabla de contenidos 2
1. Introducción 4
1.1 Objetivo 4
1.2 Alcance 4
1.3 Organización del documento 4
2. Metodología 4
3. Análisis de los casos de uso 5
3.1 Caso de uso: consultar instancias 5
3.1.1 Diagrama de caso de uso 5
3.1.2 Caso de uso 6
3.1.3 Diagrama de actividad 6
3.1.4 Diagrama robustness 7
3.1.5 Diagrama de secuencia 7
3.1.6 Diagrama de clases 8
3.2 Caso de uso: ejecutar instancia 9
3.2.1 Diagrama de caso de uso 9
3.2.2 Caso de uso 9
3.2.3 Diagrama de actividad 10
3.2.4 Diagrama robustness 11
3.2.5 Diagrama de secuencia 11
3.2.6 Diagrama de clases 12
3.3 Caso de uso: cancelar instancia 13
3.3.1 Diagrama de caso de uso 13
3.3.2 Caso de uso 13
3.3.3 Diagrama de actividad 14
3.3.4 Diagrama robustness 14
3.3.5 Diagrama de secuencia 15
3.3.6 Diagrama de clases 16
3.4 Caso de uso: retroceder instancia 17
3.4.1 Diagrama de caso de uso 17
3.4.2 Caso de uso 17
3.4.3 Diagrama de actividad 18
3.4.4 Diagrama robustness 18
3.4.5 Diagrama de secuencia 19
3.4.6 Diagrama de clases 20
3.5 Caso de uso: avanzar instancia 21
3.5.1 Diagrama de caso de uso 21
3.5.2 Caso de uso 21
3.5.3 Diagrama de actividad 22
3.5.4 Diagrama robustness 23
3.5.5 Diagrama de secuencia 23
3.5.6 Diagrama de clases 25
3.6 Caso de uso: rellenar campo 26
3.6.1 Diagrama de caso de uso 27
3.6.2 Caso de uso 27
3.6.3 Diagrama de actividad 28
3.6.4 Diagrama robustness 28
3.6.5 Diagrama de secuencia 29
3.6.6 Diagrama de clases 29
4. Diseño de la arquitectura 30

5. Diseño de la Base de datos....31
5.1 Definición de tablas.... 31
5.1.1 Orh_Wfl_Workflow y Orh_Wfl_WorkFlowAreas.... 31
5.1.2 Orh_Wfl_WorkflowStep y Orh_Wfl_WorkflowField.... 32
5.1.3 Orh_Wfl_WorkflowInstanceField 33
5.2 Modelo relacional 34
6. Bibliografía....35

1. INTRODUCCIÓN

1.1 Objetivo

El presente documento tiene por objetivo realizar un análisis detallado de los casos de uso y de los requisitos de la aplicación y con él diseñar la estructura de clases y datos del sistema.

1.2 Alcance

Se analizarán en profundidad todos los casos de uso del sistema y se detallarán con ayuda de diagramas. Los nuevos casos de uso y requisitos que puedan surgir del análisis se añadirán a los ya existentes. Tras el análisis se procederá al diseño de la aplicación.

1.3 Organización del documento

El presente documento contiene las siguientes secciones:

1. Introducción: Descripción del objetivo y alcance del documento.
2. Breve descripción de la metodología utilizada para el análisis y diseño.
3. Análisis de los casos de uso.
4. Diseño de la arquitectura.
5. Diseño de la base de datos.
6. Bibliografía: Referencias a libros y webs utilizadas en el documento.

2. METODOLOGÍA

Se va a hacer un análisis y diseño orientado a objetos (OOAD) [1]. El objetivo es obtener el diseño del sistema, reflejado en el modelo de clases. Para ello se van a seguir estos pasos:

1. Se toma como base los casos de uso y el modelo de negocio.
2. Mediante diagramas de robustez (*robustness*)[1] se convierten en diagramas de secuencia:

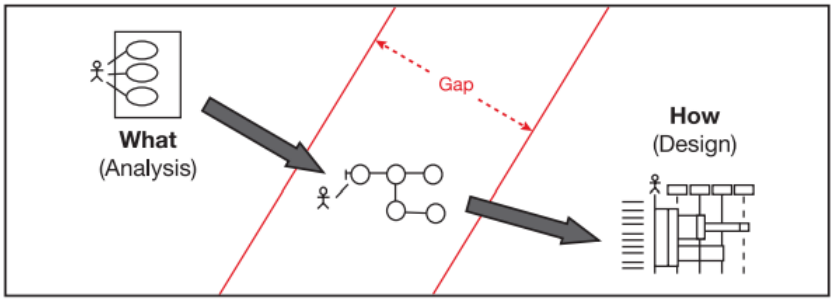


Figura 1: Diagramas robustness

3. A partir de los diagramas de secuencia se diseña el modelo de clases.

Además del diseño del sistema, se tiene que diseñar la arquitectura del sistema. Para ello, se debe tener en consideración el modelo de clases obtenido y los requisitos no funcionales, ya que algunos de estos requisitos condicionan la arquitectura.

---

[1] Los diagramas *robustness* no forman parte de UML pero son una herramienta muy útil durante la fase de análisis para transformar casos de uso en diagramas de secuencia.

3. ANÁLISIS DE LOS CASOS DE USO

En este apartado se va a realizar el análisis de los casos de uso del sistema a partir del estudio realizado en el documento de modelo de negocio. A partir de ellos, se identificarán otros procesos que tienen lugar en el sistema y que pueden no estar reflejados en los casos de uso, se analizarán, se ofrecerán otros puntos de vista mediante diferentes diagramasy por último se detallará un modelo de clases que pueda servir de base para la posterior etapa de diseño de la aplicación.

Los diagramas que se van a usar para este análisis son:

- Diagramas de casos de uso: describen la relación entre los actores (o usuarios del sistema) y los casos de uso.
- Diagramas de actividad: describen el flujo de trabajo (en ocasiones también el flujo de datos) del caso de uso.
- Diagramas *robustness*: sirven de puente entre los casos de uso y los diagramas de secuencia.
- Diagramas de secuencia: muestran la vista dinámica del caso de uso, las interacciones entre los componentes.
- Diagramas de clases: describen el sistema bajo la perspectiva de la orientación a objetos (OO).

3.1 Caso de uso: consultar instancias

A partir del caso de uso identificado en la sección 2.2 del Modelo de negocio [2], se realiza un análisis más detallado y se identifican nuevos casos de uso de filtrado.

3.1.1 Diagrama de caso de uso

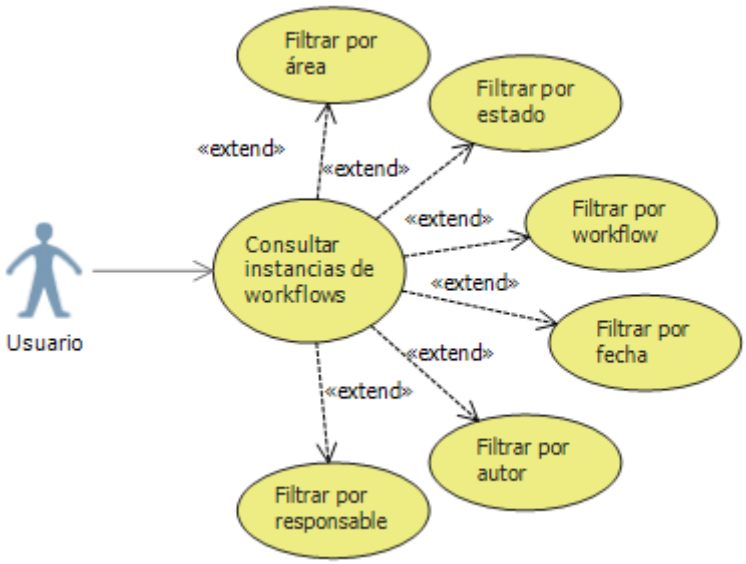


Figura 2: Diagrama caso de uso Consultar instancias

3.1.2 Caso de uso

Caso de uso:	Consultar instancias
Descripción:	El usuario hace una consulta sobre los workflows seleccionado diferentes filtros de búsqueda. Nota: Las consultas las puede realizar cualquier usuario de WfMS (no tiene por qué ser responsable de algún workflow).
Actores:	Usuario de workflows.
Pre condiciones:	El usuario está autentificado en el sistema.
Flujo normal:	1. El usuario selecciona uno o más de los posibles filtros de búsqueda (estado, workflow, fecha, autor o participante) 2. El sistema le devuelve las instancias que cumplen los filtros indicados (puede no devolver ninguna).
Flujo alternativo:	-
Post condiciones:	-

3.1.3 Diagrama de actividad

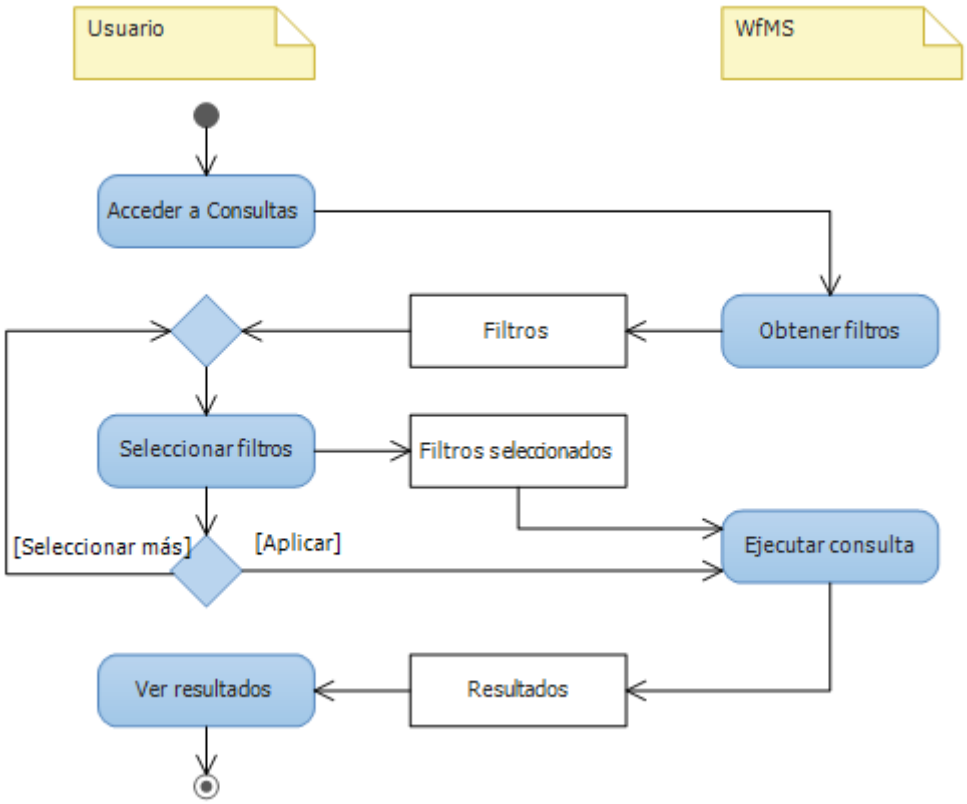


Figura 3: Diagrama de actividad del caso de uso Consultar instancias

3.1.4 Diagrama robustness

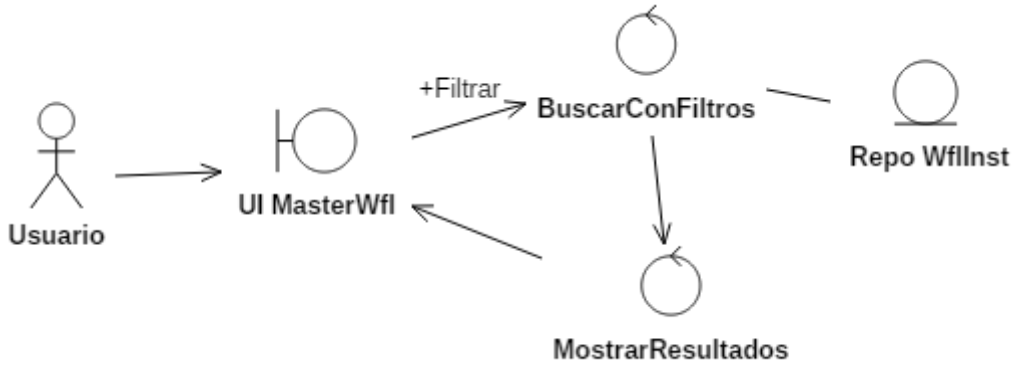


Figura 4: Diagrama robustness de Consultar instancias

3.1.5 Diagrama de secuencia

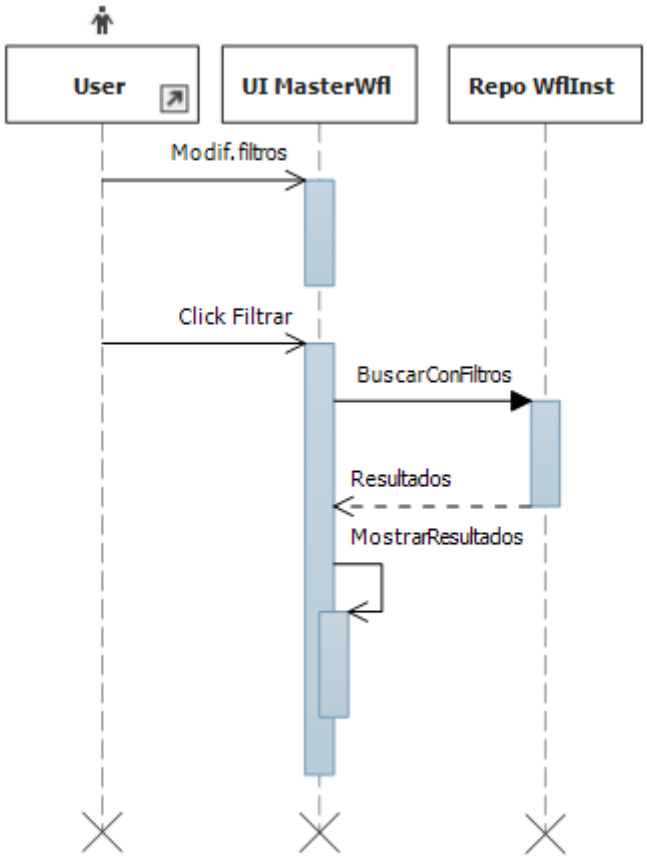


Figura 5: Diagrama de secuencia del caso de uso Consultar instancias

3.1.6 Diagrama de clases

A continuación, se muestra el diagrama de clases que se ha desarrollado a partir de este caso de uso. Se han definido los siguientes estados para los workflows y para los steps:

- Activo.
- Cancelado.
- Finalizado.
- No iniciado: sólo aplica a steps. Cuando un paso es rechazado, el paso anterior se marca como "Activo" y éste se marca como "No iniciado".

En la clase "Area" se ha añadido una referencia a sí mismo para permitir jerarquía de áreas.

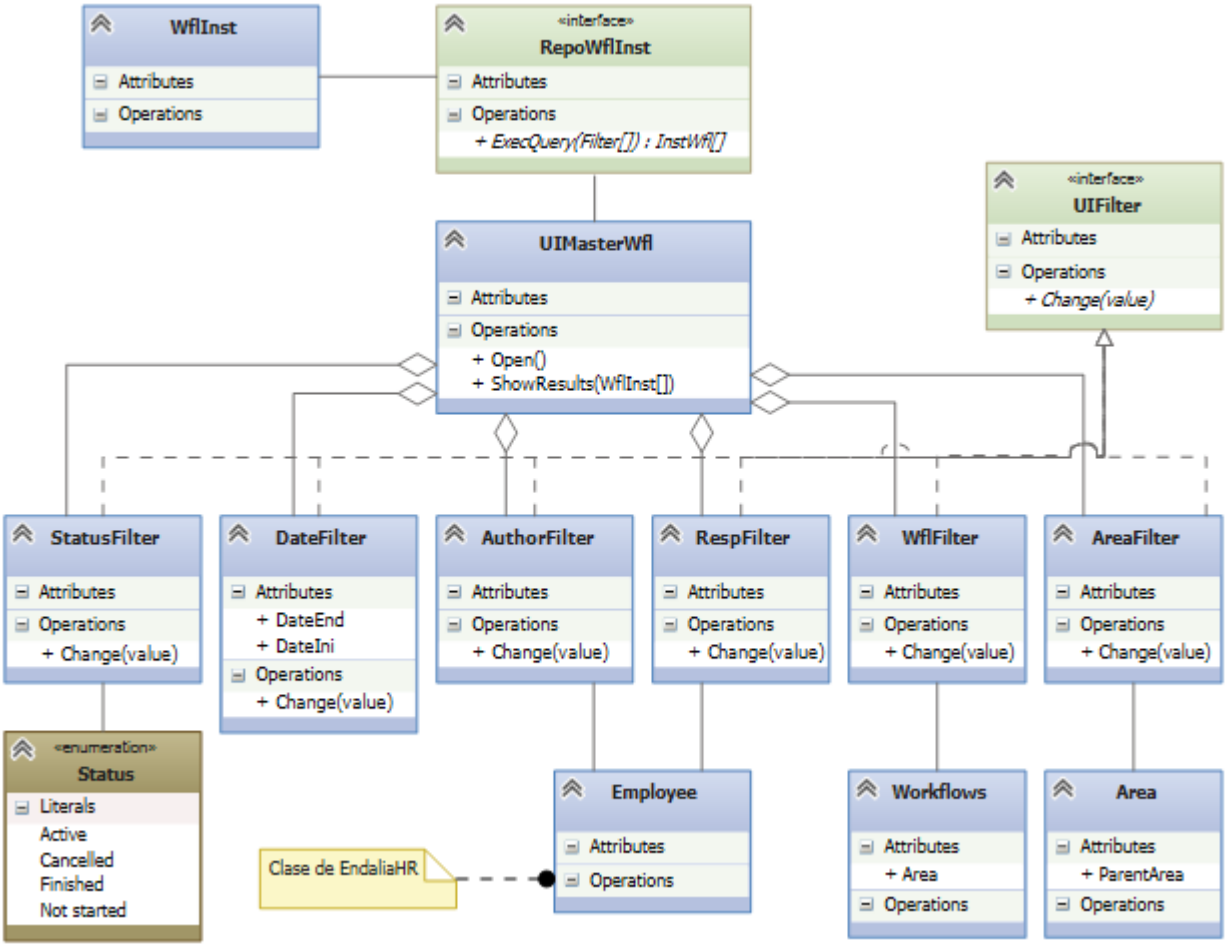


Figura 6: Diagrama de clases caso de uso Consultar workflows

3.2 Caso de uso: ejecutar instancia

3.2.1 Diagrama de caso de uso

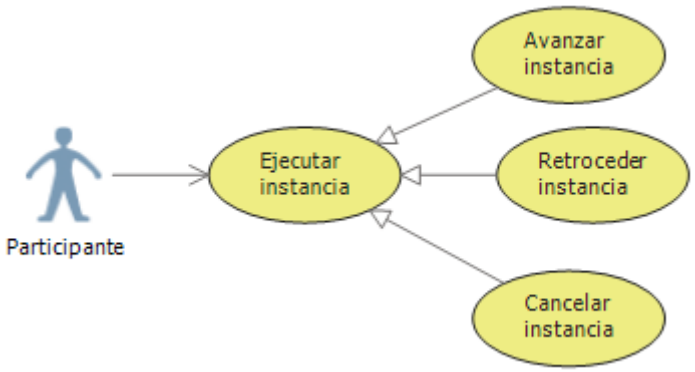


Figura 7: Diagrama de caso de uso Ejecutar instancia

3.2.2 Caso de uso

Caso de uso:	Ejecutar instancia
Descripción:	El participante accede a la interfaz de usuario para ejecutar un paso de workflow.
Actores:	Participante de workflows.
Pre condiciones:	La instancia que quiere ejecutar tiene al menos un paso en ejecución. El participante es responsable del paso activo.
Flujo normal:	1. El usuario accede a la UI. 2. El sistema le muestra los campos de información del paso así como los botones de acción disponibles (avanzar, rechazar o cancelar).
Flujo alternativo:	Si el participante no es responsable del paso actual, el sistema le muestra un mensaje de aviso y no le deja acceder a la información del workflow.
Post condiciones:	-

3.2.3 Diagrama de actividad

Para entender el comportamiento de este caso de uso (avanzar o retroceder un flujo) es necesario introducir un nuevo concepto, *nivel*, que hace referencia al orden cronológico en que se ejecutan los pasos de un workflow. En el siguiente esquema queda reflejada la relación entre los niveles y los pasos, tanto para workflows lineales como para no lineales:

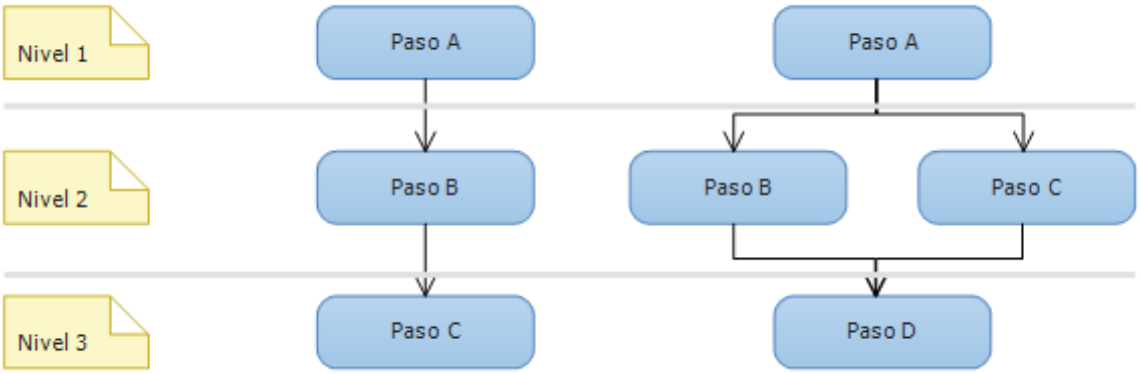


Figura 8: Niveles de los pasos de un workflow

A continuación, el diagrama de actividad para los 3 casos de uso (avanzar, retroceder y cancelar). Nótese que "Paso N" hace referencia al paso de nivel N, tal cual se acaba de explicar:

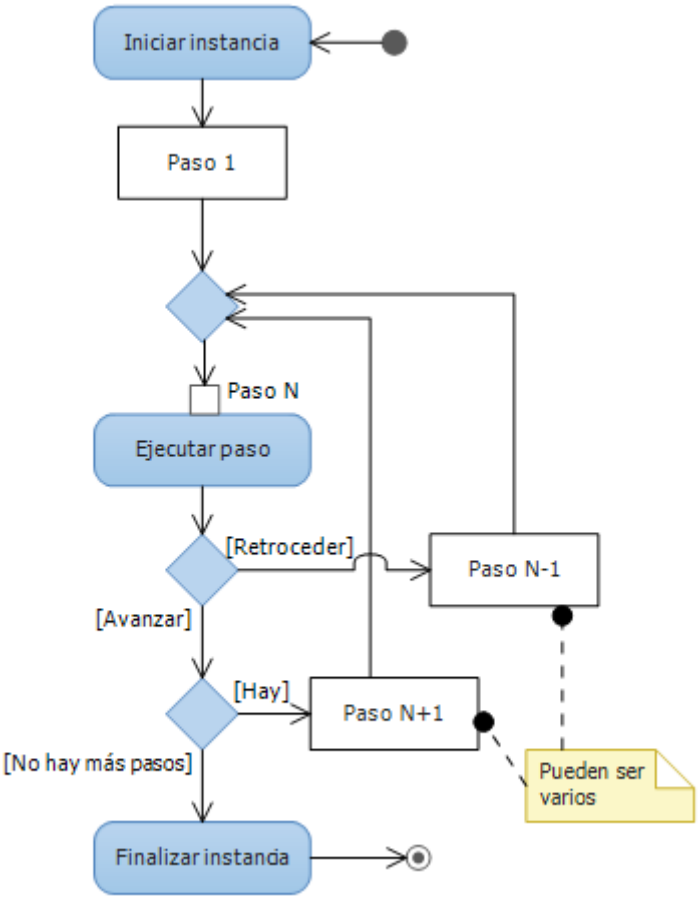


Figura 9: Diagrama de actividad del caso de uso Ejecutar paso

3.2.4 Diagrama robustness

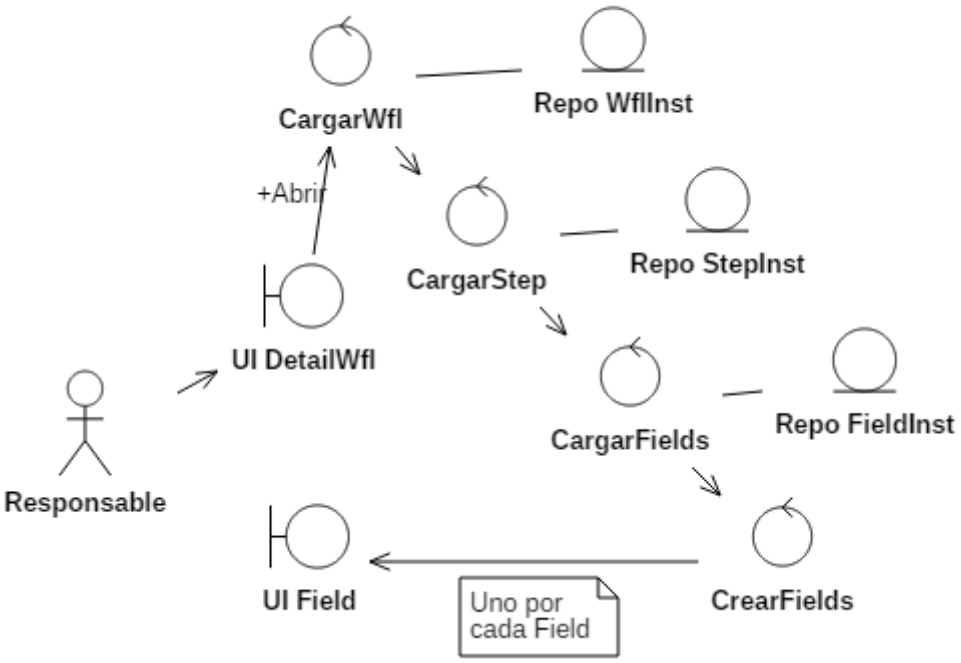


Figura 10: Diagrama robustness de Ejecutar instancia

3.2.5 Diagrama de secuencia

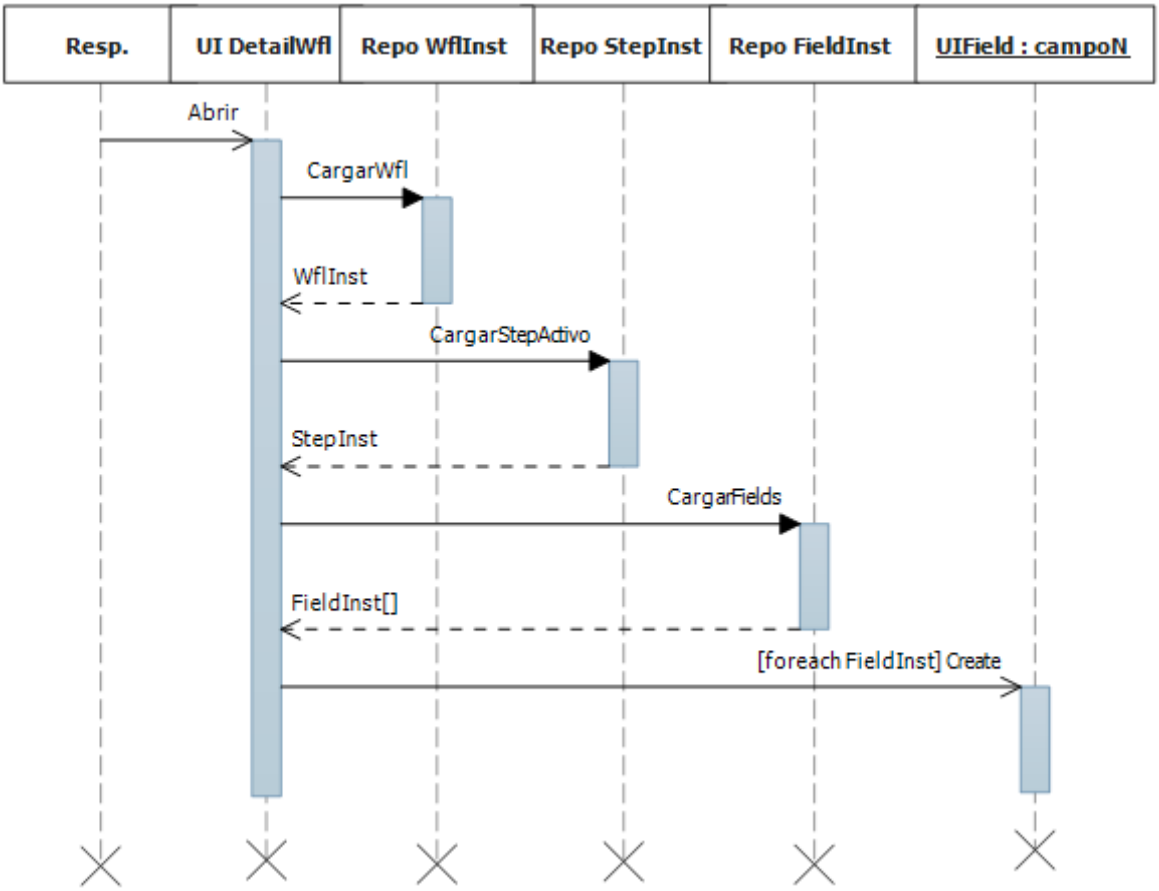


Figura 11: Diagrama de secuencia del caso de uso Ejecutar instancia

3.2.6 Diagrama de clases

A continuación, se muestra el diagrama de clases que se ha desarrollado a partir de este caso de uso. En la entidad Instancia de workflow, se ha añadido un atributo llamado “KeyName” que servirá para identificar al objeto y al ser de tipo cadena, será más cómodo que un identificador de tipo numérico de cara al usuario.
Se han tenido en cuenta los requisitos que permiten que un campo sea obligatorio, de sólo lectura o pueda ocultarse según un determinado criterio. También se han indicado los tipos de responsable de un paso que están definidos en los requisitos funcionales.

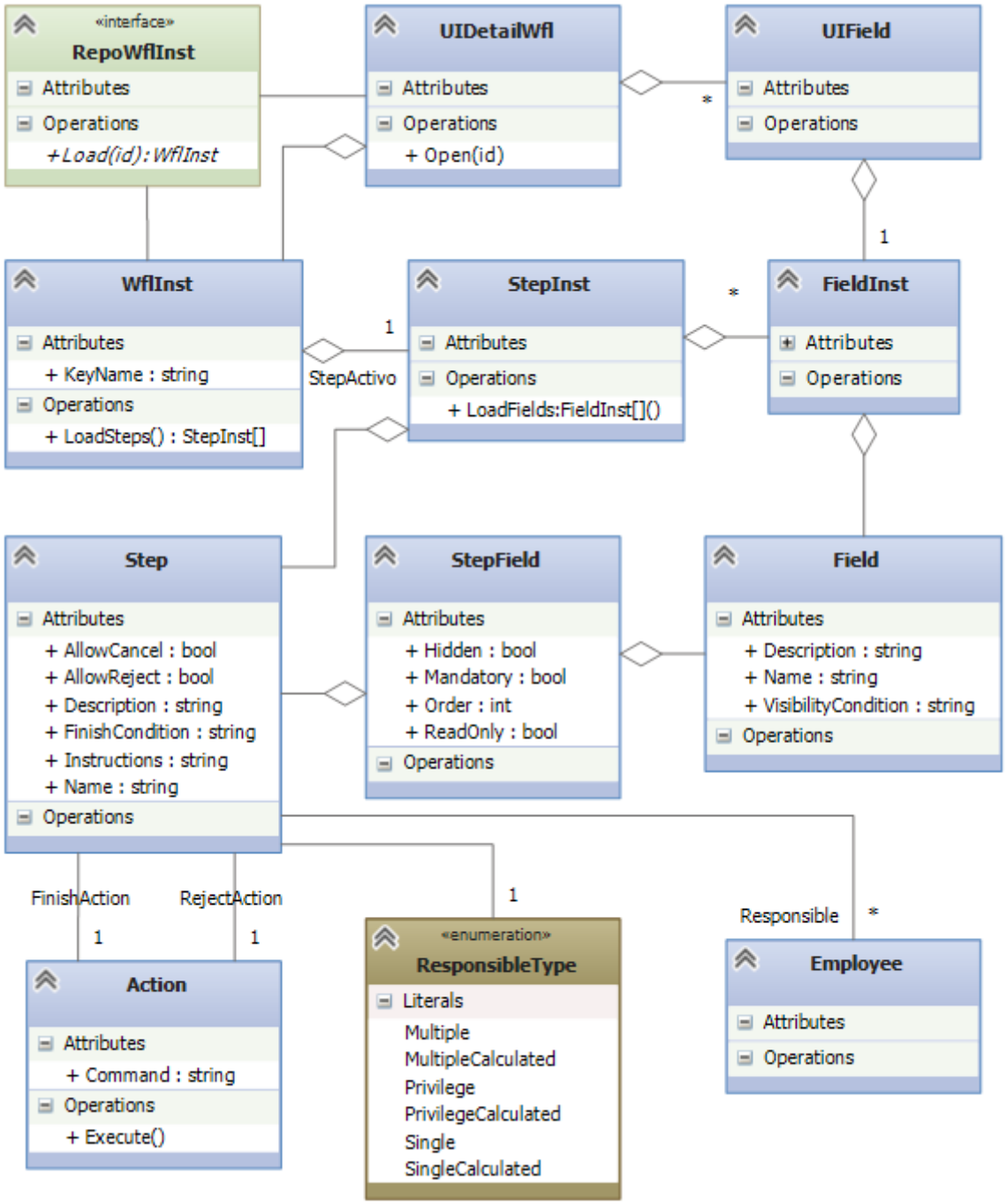


Figura 12: Diagrama de clases Ejecutar paso

3.3 Caso de uso: cancelar instancia

Tras el análisis de este caso de uso del Modelo de negocio, se especifica como requisito que las notificaciones van a ser vía email (notificaciones por otra vía quedan fuera del alcance de este proyecto). Además, se incluye un nuevo actor, ya que el destinatario de una notificación no tiene por qué ser un usuario del WfMS, puede ser un destinatario externo.

3.3.1 Diagrama de caso de uso

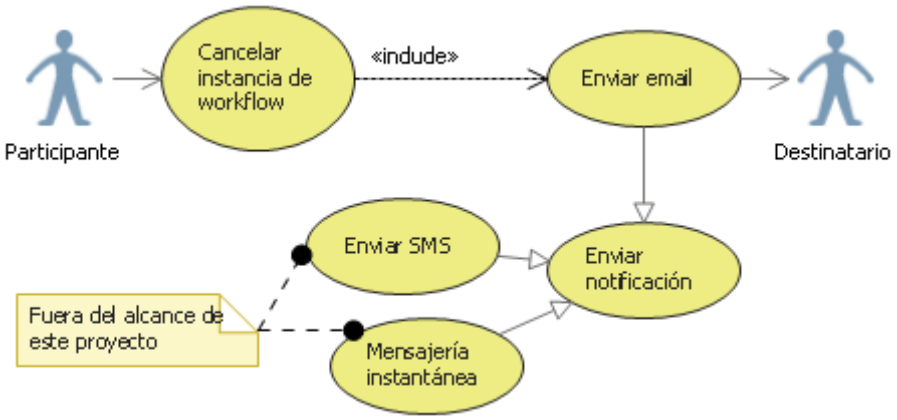


Figura 13: Caso de uso Cancelar instancia

3.3.2 Caso de uso

Caso de uso:	Cancelar instancia
Descripción:	El participante cancela la ejecución de una instancia.
Actores:	Participante de workflows. Destinatario de la notificación.
Pre condiciones:	La instancia está activa (no ha sido cancelada previamente). La configuración del workflow permite cancelarlo desde la fase actual.
Flujo normal:	1. El participante cancela la instancia. 2. El sistema pone la instancia en estado cancelado (no podrá avanzarse ni retrocederse). 3. El sistema notifica a los destinatarios que el flujo ha sido cancelado.
Flujo alternativo:	-
Post condiciones:	La instancia está en estado cancelado.

3.3.3 Diagrama de actividad

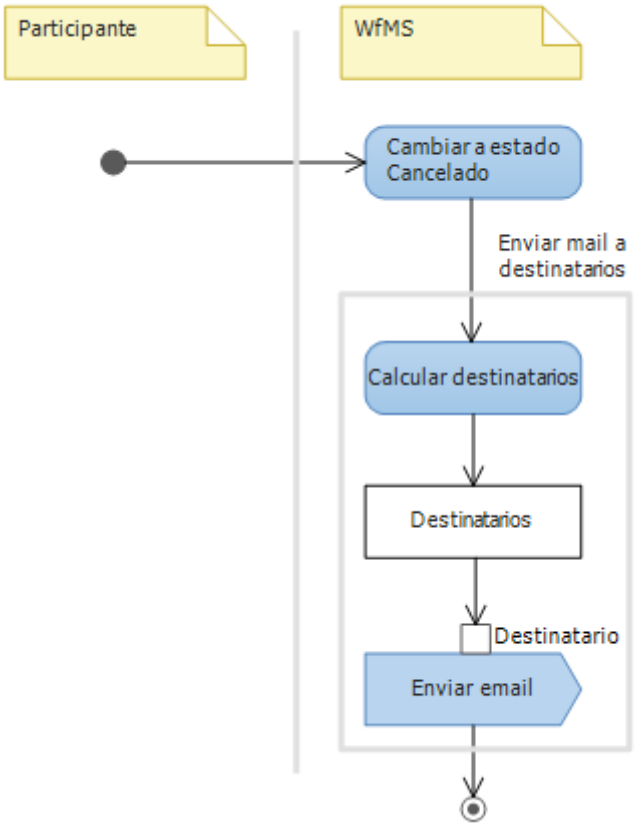


Figura 14: Diagrama de actividad del caso de uso Cancelar instancia

3.3.4 Diagrama robustness

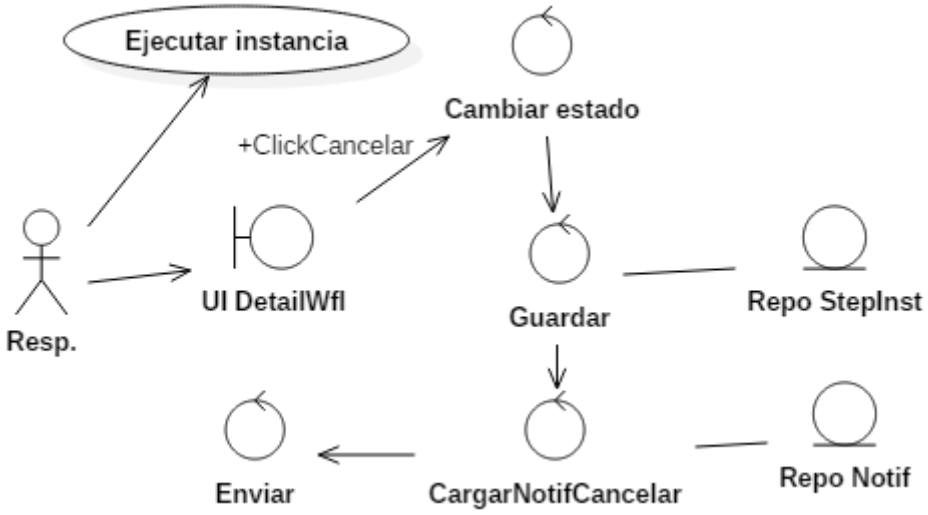


Figura 15: Diagrama robustness Cancelar instancia

3.3.5 Diagrama de secuencia

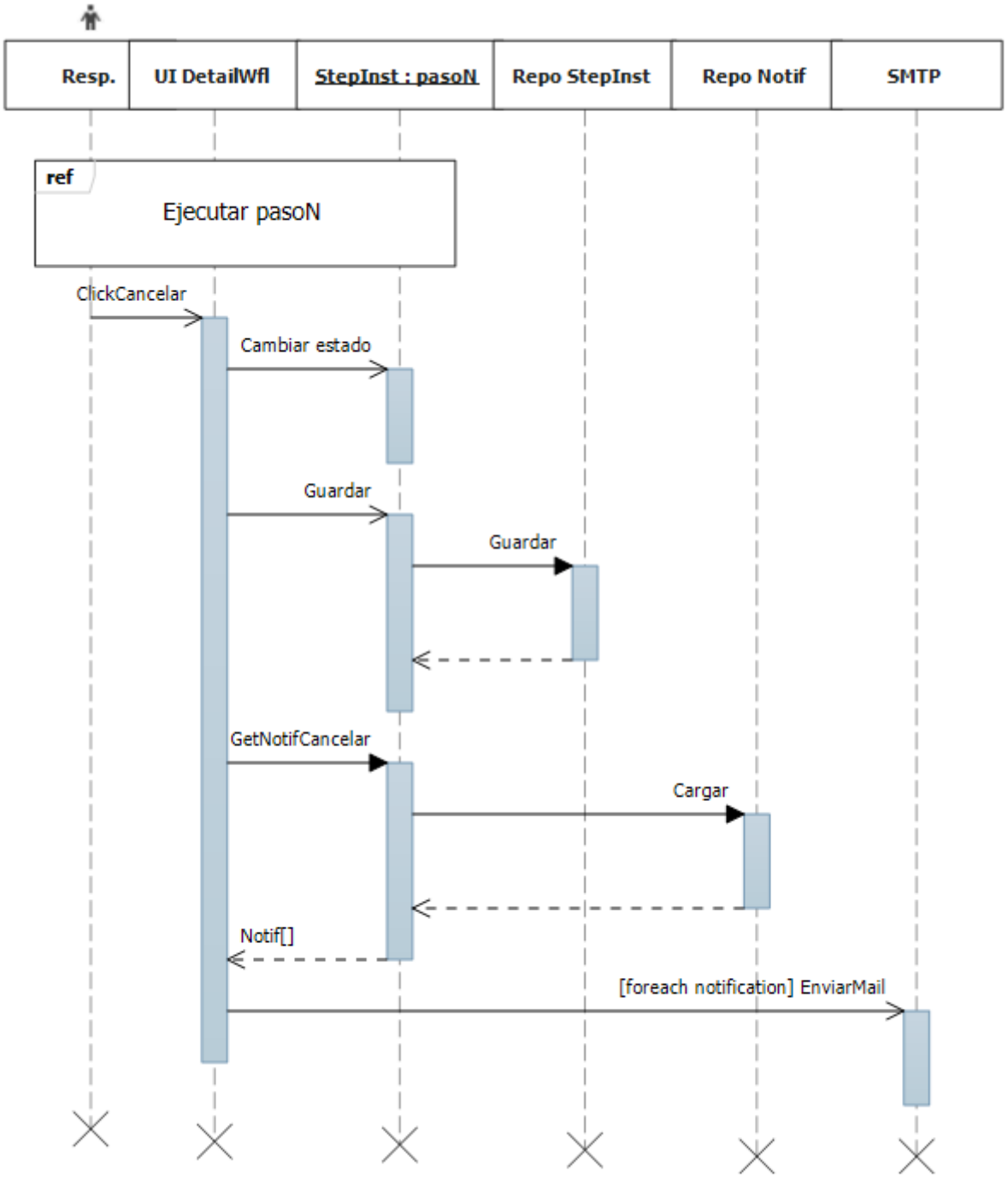


Figura 16: Diagrama de secuencia de caso de uso Cancelar instancia

3.3.6 Diagrama de clases

A continuación, se muestra el diagrama de clases que se ha desarrollado a partir de este caso de uso.

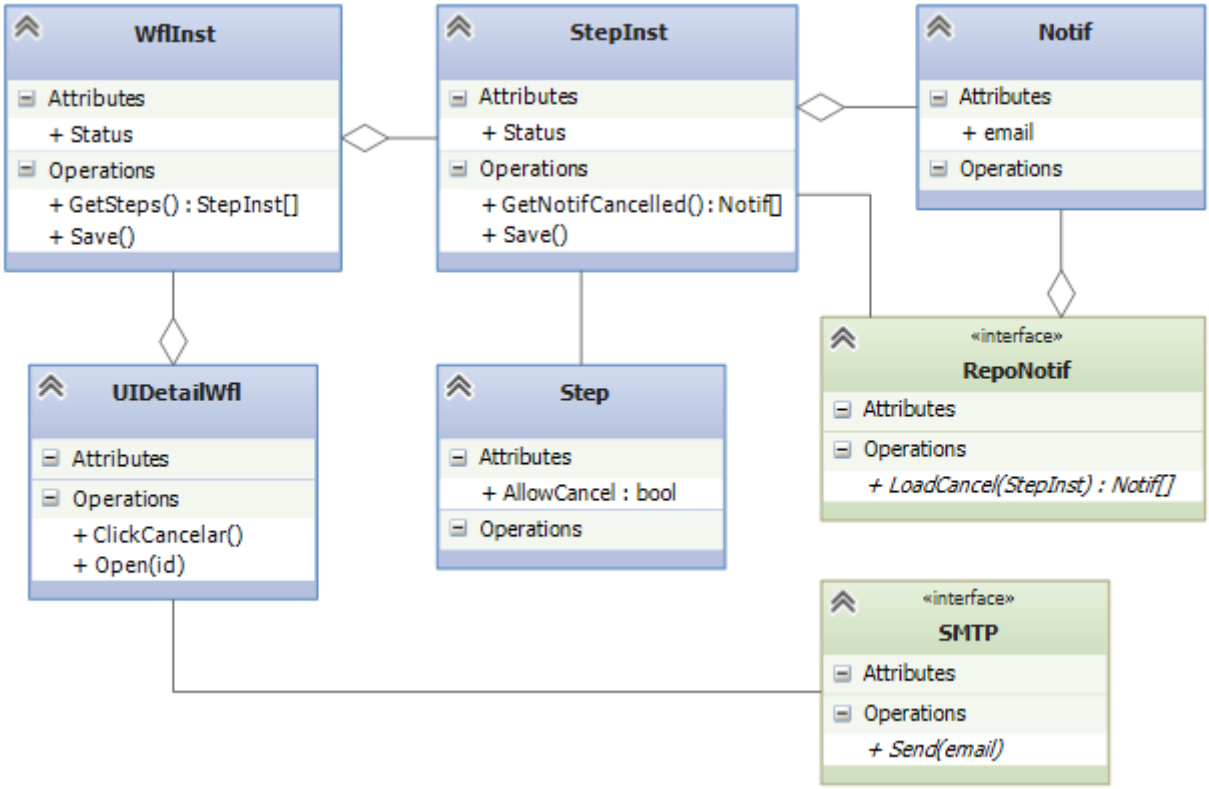


Figura 17: Diagrama de clases Cancelar instancia

3.4 Caso de uso: retroceder instancia

3.4.1 Diagrama de caso de uso

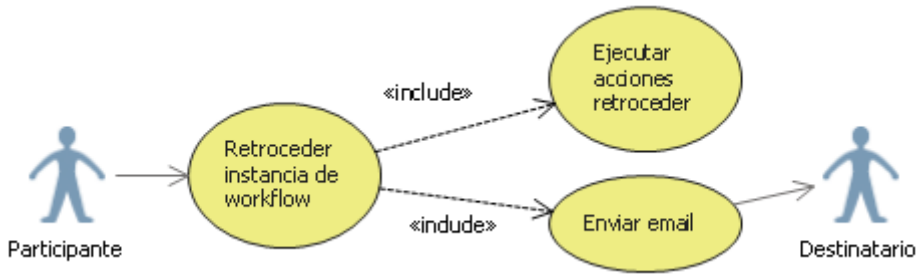


Figura 18: Diagrama de caso de uso Retroceder instancia

3.4.2 Caso de uso

Caso de uso:	Retroceder instancia
Descripción:	El participante retrocede la instancia al paso o pasos anteriores.
Actores:	Participante de workflows. Destinatario de la notificación.
Pre condiciones:	Existe paso o pasos anteriores. La configuración del workflow permite retroceder desde el paso actual.
Flujo normal:	1. El participante retrocede la instancia. 2. El sistema cambia el estado del paso a “No iniciado” (distinto de “Finalizado”). 3. El sistema ejecuta las acciones de retroceso del paso. 4. El sistema obtiene los pasos anteriores y los pone en estado “Activo”. 5. El sistema notifica a los destinatarios que el flujo ha sido retrocedido.
Flujo alternativo:	-
Post condiciones:	La instancia está el paso o pasos anteriores.

3.4.3 Diagrama de actividad

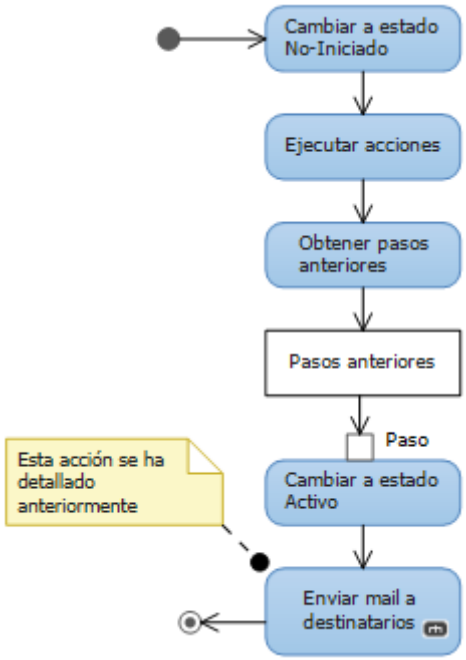


Figura 19: Diagrama de actividad del caso de uso Retroceder instancia

3.4.4 Diagrama robustness

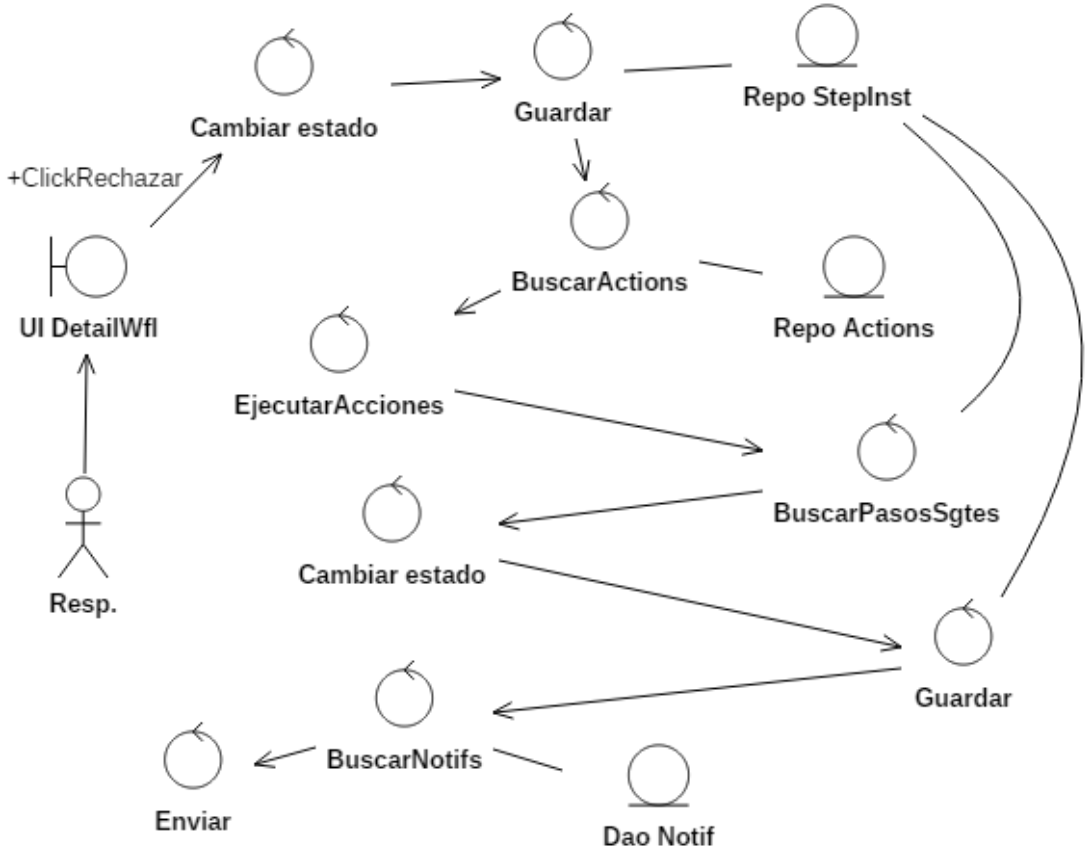


Figura 20: Diagrama robustness de Rechazar paso

3.4.5 Diagrama de secuencia

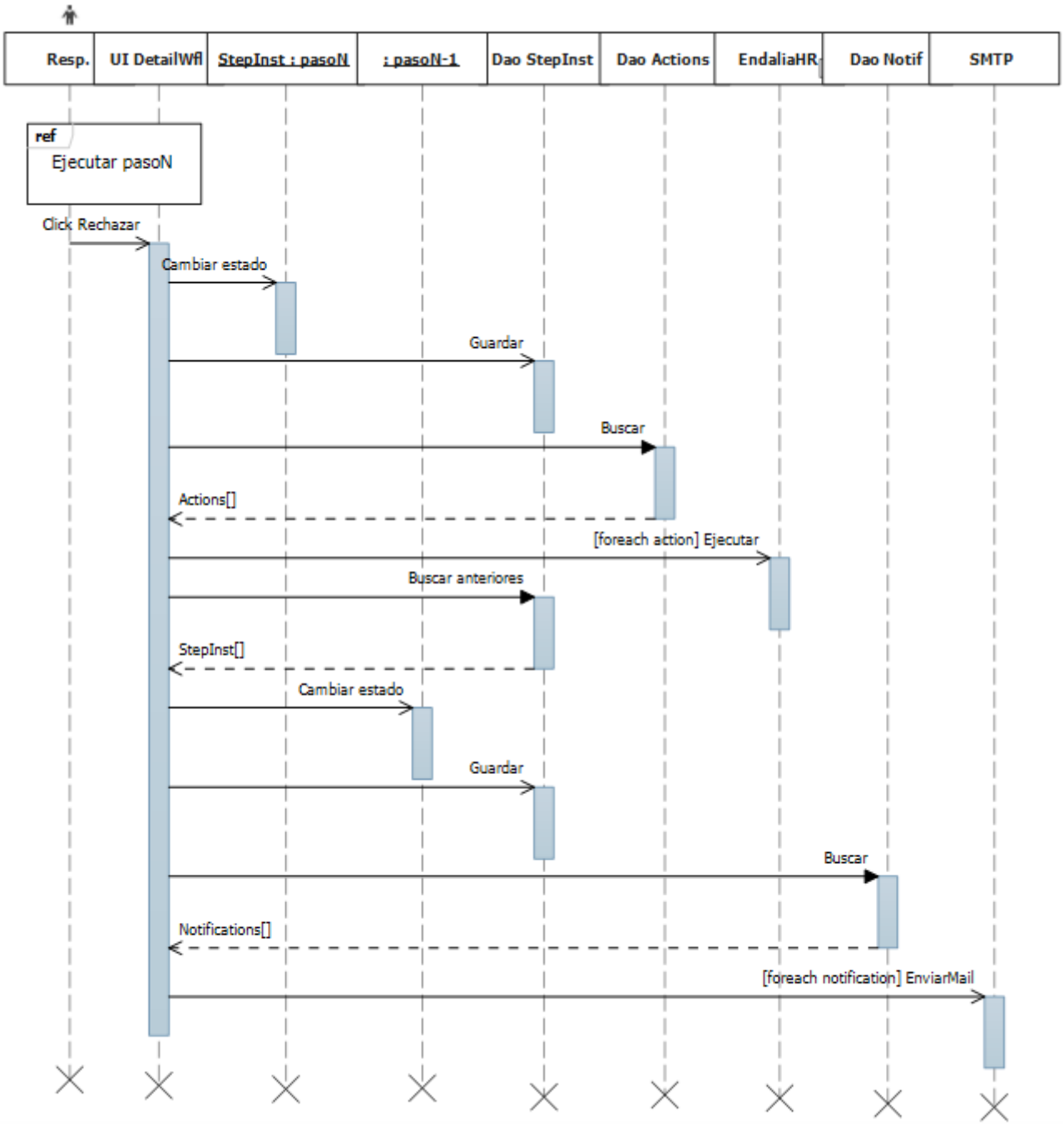


Figura 21: Diagrama de secuencia del caso de uso Retroceder paso

3.4.6 Diagrama de clases

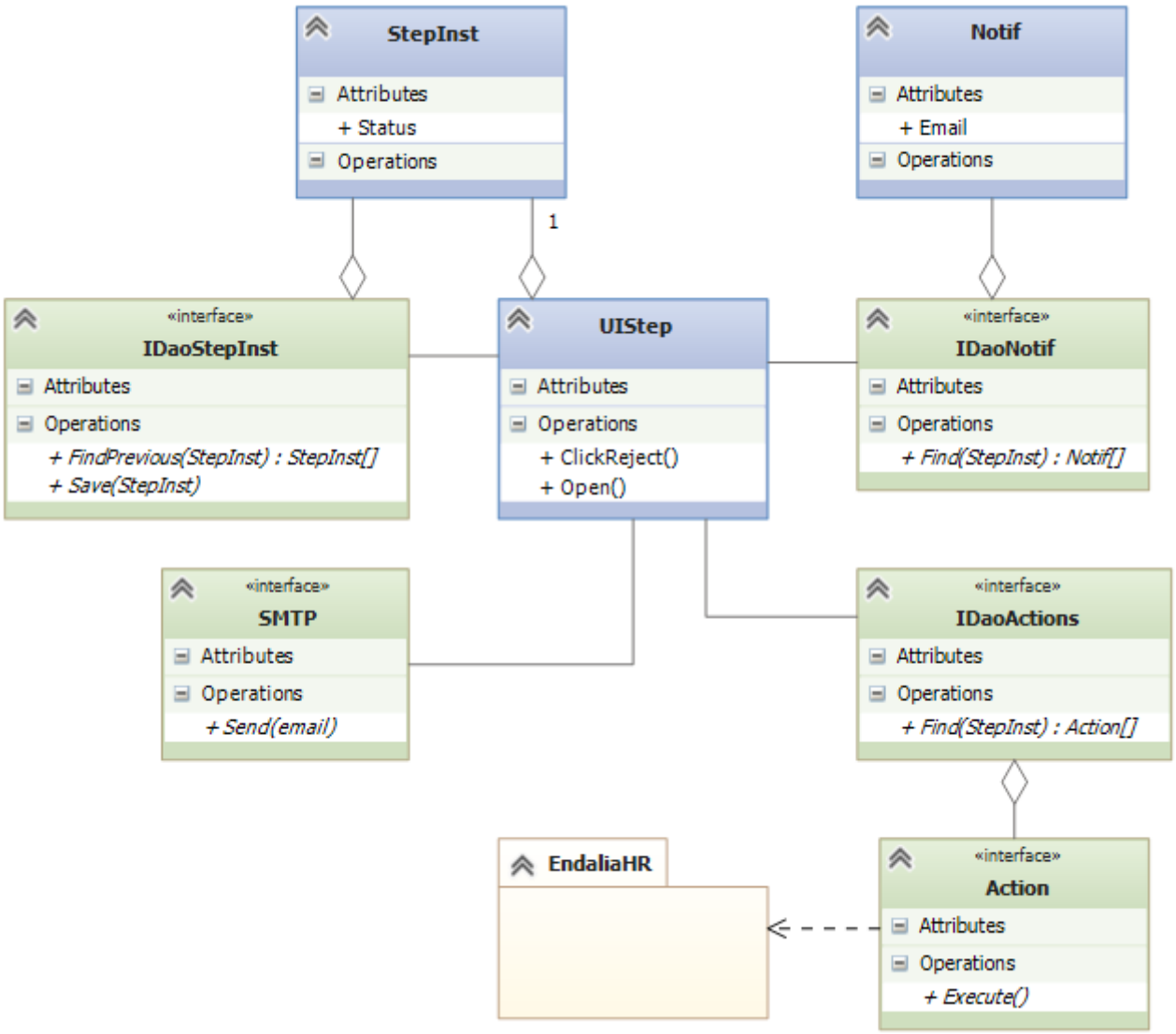


Figura 22: Diagrama de clases Rechazar paso

3.5 Caso de uso: avanzar instancia

El análisis de este caso de uso lo completa con nueva funcionalidad:

- Comprobar condiciones: se pueden configurar determinadas condiciones que deben cumplirse para poder avanzar de paso. Un ejemplo de uso de esta funcionalidad sería: comprobar en un workflow de alta de empleado que el DNI introducido no existe en el sistema.
- Un caso particular de la funcionalidad anterior es comprobar si falta algún campo por rellenar. Para ello se añade un nuevo requisito funcional RF-19 que indica que los campos pueden ser obligatorios.
- Del mismo modo que un campo puede ser obligatorio, se añade otro requisito RF-20 que indique que los campos de información, pueden ser de sólo lectura en algunos pasos del workflow.
- Un campo puede ser no visible en algunos pasos del workflow. Esta funcionalidad se recoge en el requisito RF-21

3.5.1 Diagrama de caso de uso

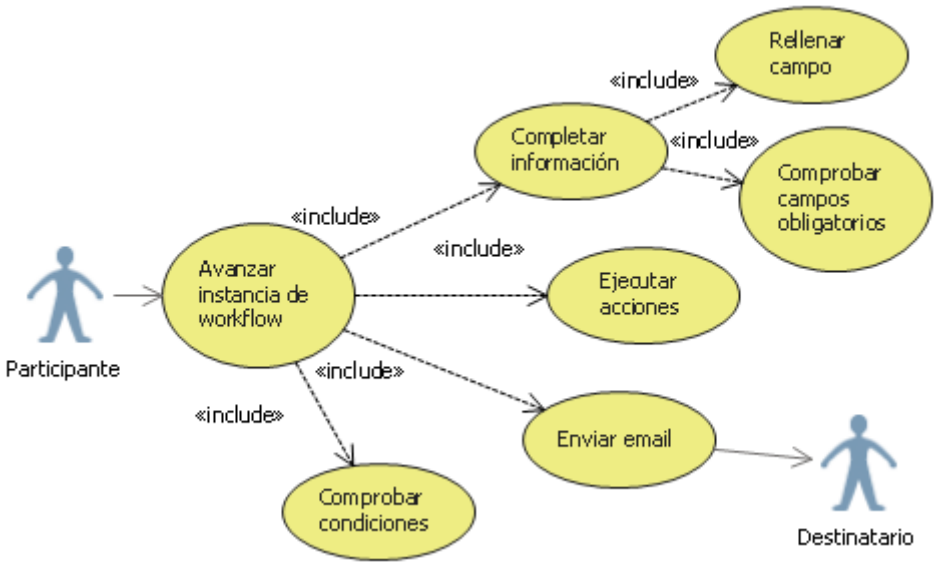


Figura 23: Diagrama de caso de uso de avanzar instancia

3.5.2 Caso de uso

Caso de uso:	Avanzar instancia
Descripción:	El usuario avanza el workflow a la siguiente fase.
Actores:	El participante del paso en ejecución. Los destinatarios de las notificaciones.
Pre condiciones:	Hay pasos siguientes.
Flujo normal:	1. El usuario completa la información. 2. El sistema comprueba que se han rellenado los campos obligatorios 3. El sistema comprueba que se cumplan las condiciones de finalización del paso (si existen).

4. El sistema cambia el paso a estado “Finalizado”
5. El sistema ejecuta las acciones correspondientes.
6. El sistema calcula los siguientes pasos y los crea en estado “Activo”.
7. El sistema envía mails a los destinatarios.

Flujo alternativo:	Si el usuario no completa toda la información, el sistema le avisa de que faltan datos y no le permite avanzar. Si no se cumplen las condiciones de finalización, el sistema avisa al usuario y no le permite avanzar.
Post condiciones:	Los siguientes pasos en la secuencia están activos.

3.5.3 Diagrama de actividad

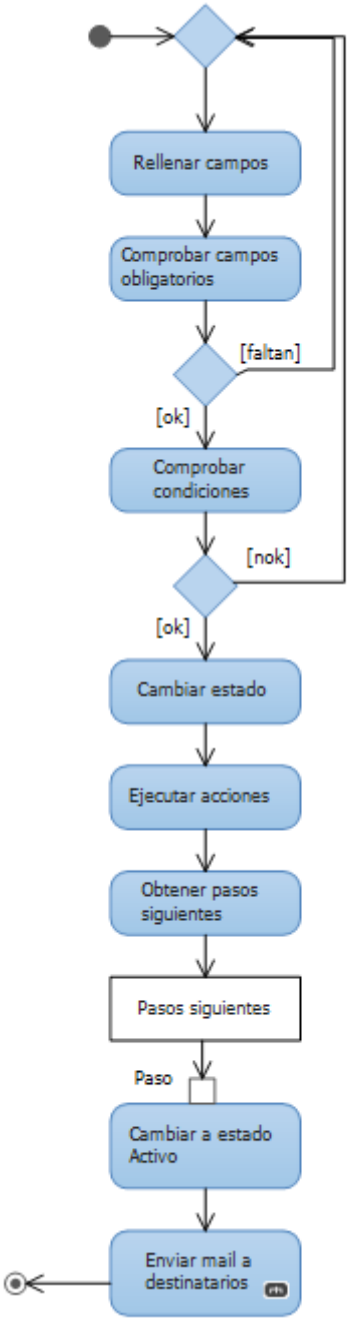


Figura 24: Diagrama de actividad del caso de uso Avanzar paso

3.5.4 Diagrama robustness

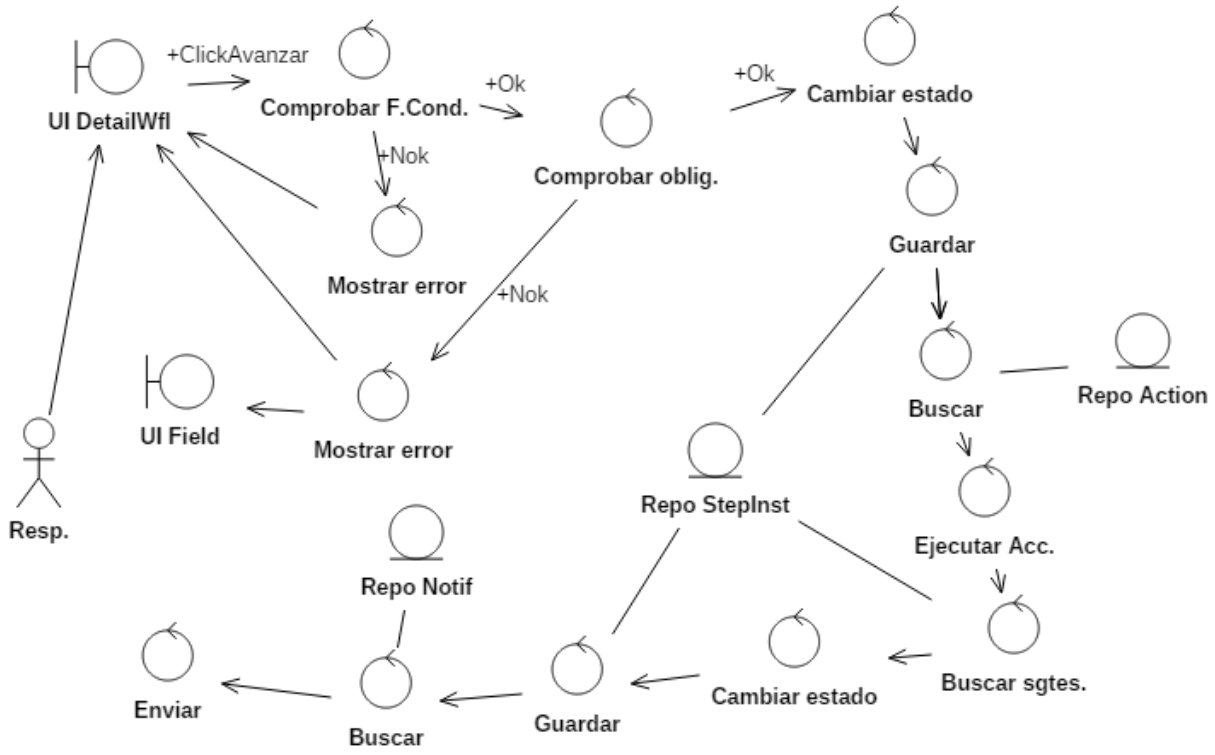


Figura 25: Diagrama robustness Avanzar paso

3.5.5 Diagrama de secuencia

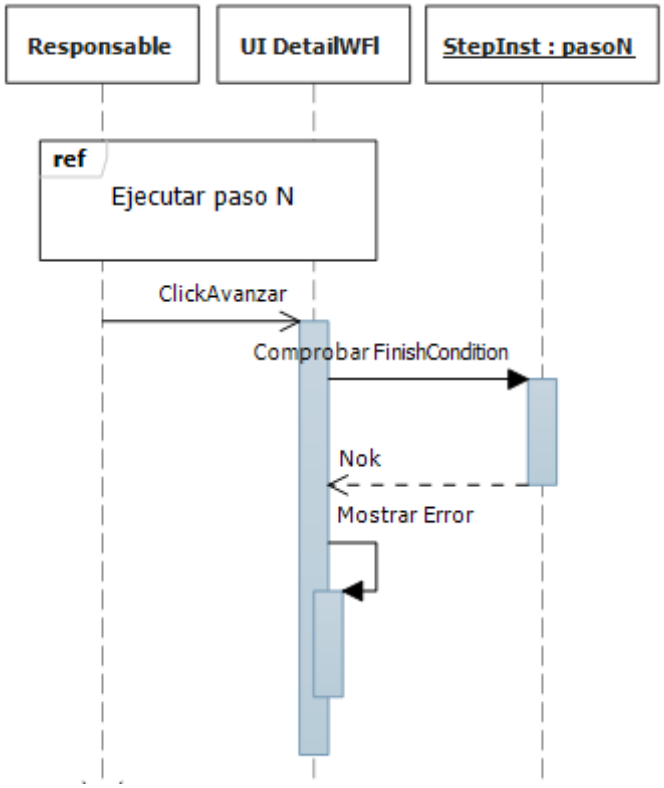


Figura 26: Diagrama de secuencia no se cumple FinishCondition

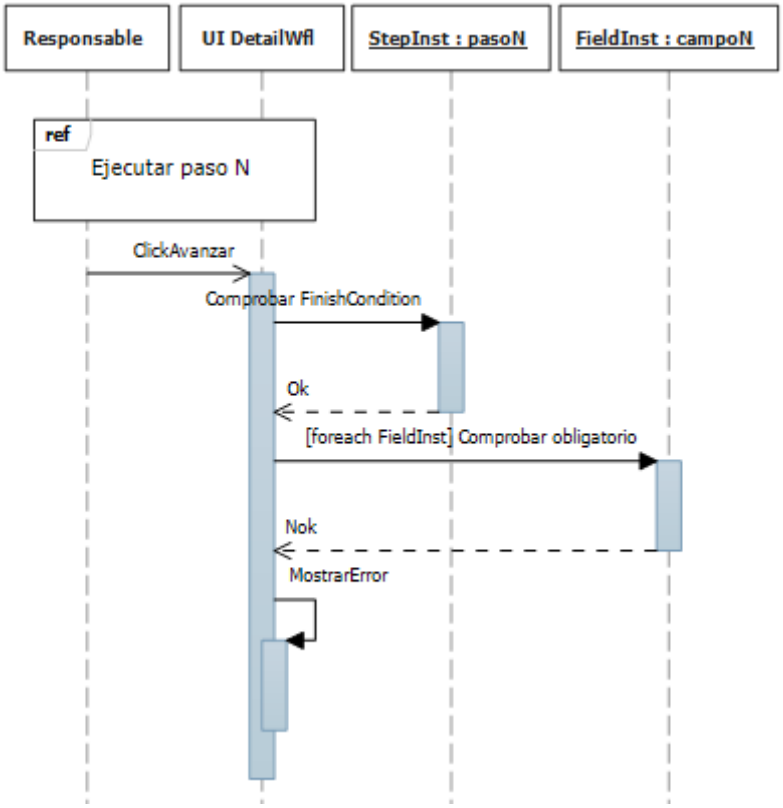


Figura 27: Diagrama de secuencia campos obligatorios sin rellenar

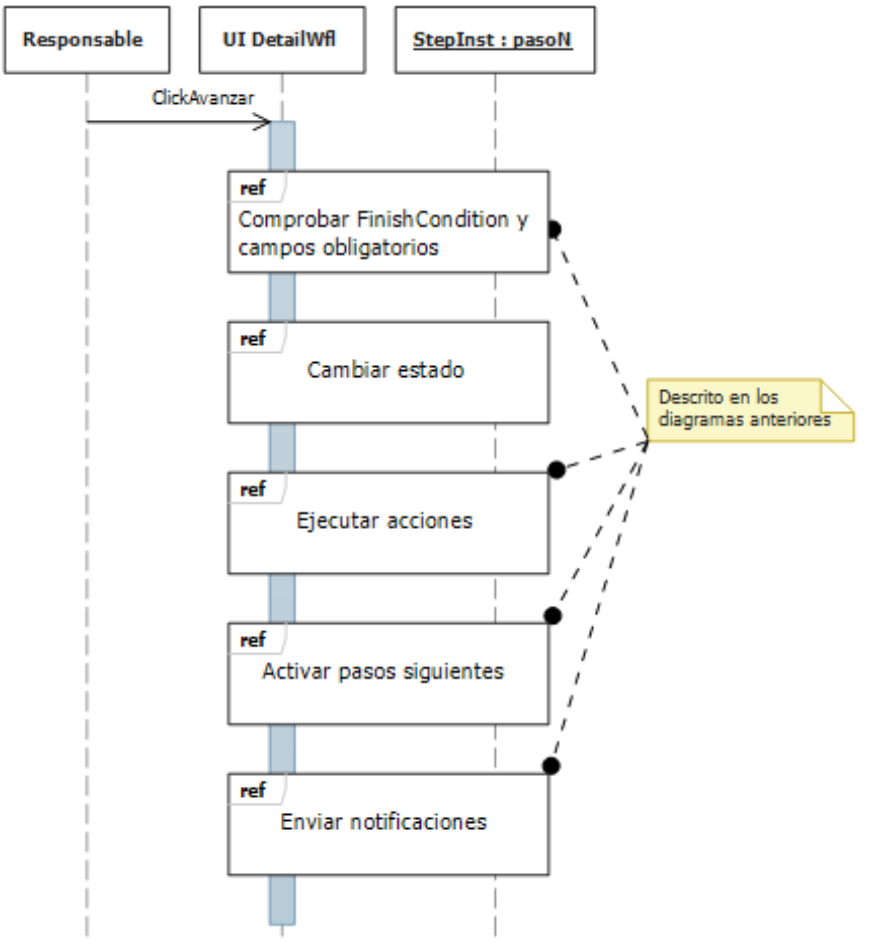


Figura 28: Diagrama de secuencia de Avanzar paso

3.5.6 Diagrama de clases

A continuación, se muestra el diagrama de clases que se ha desarrollado a partir de este caso de uso. Las acciones asociadas a un Step, se ejecutarán contra EndaliaHR, que es el sistema donde reside la lógica de negocio. Aprovechando que el WfMS está integrado en él como un módulo más y comparte base de datos con él, las acciones se ejecutarán como un procedimiento almacenado en base de datos.

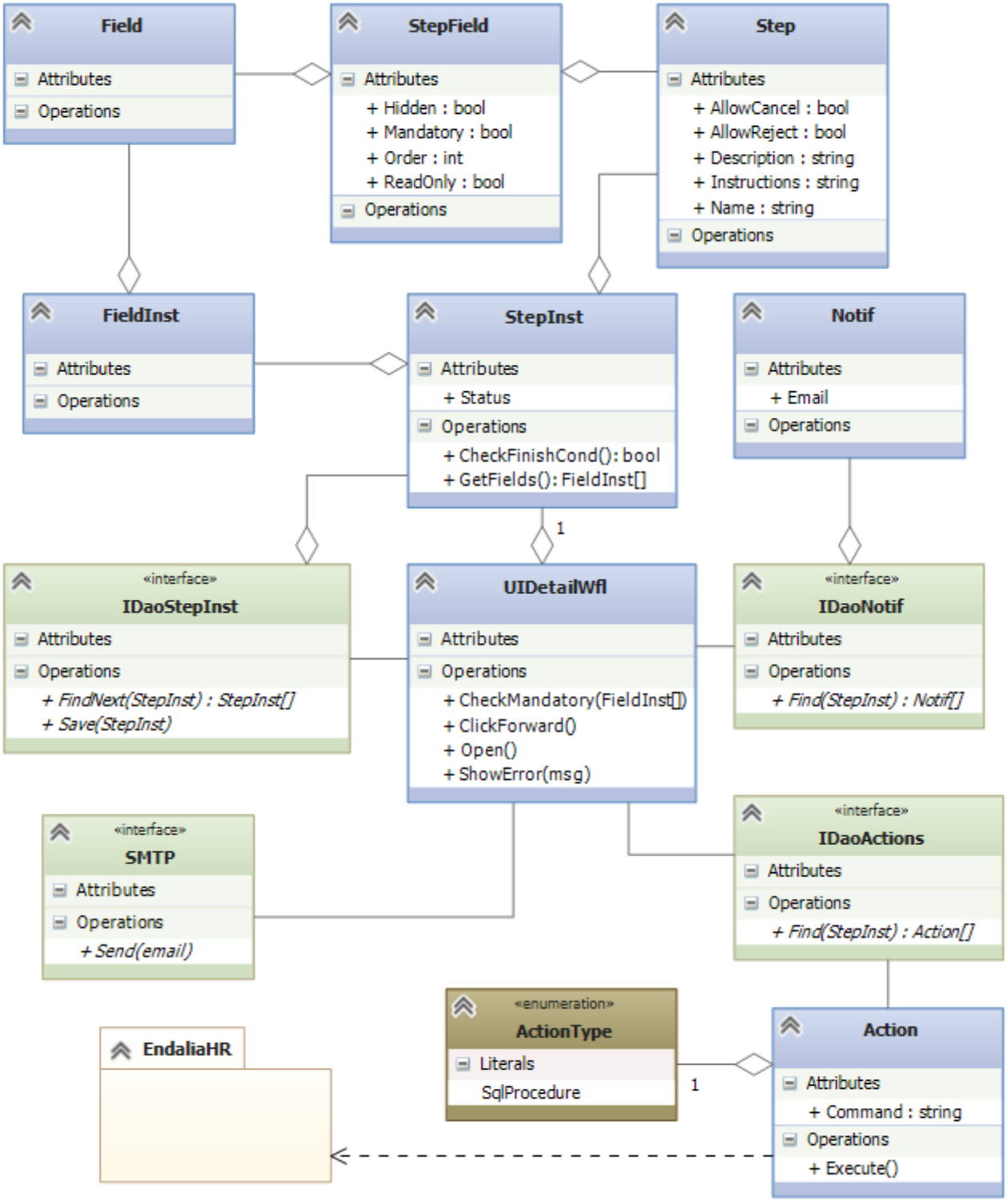


Figura 29: Diagrama de clases caso de uso avanzar instancia

3.6 Caso de uso: rellenar campo

En este apartado se analiza en profundidad el caso de uso cuando un participante rellena la información de un campo. Los campos guardan información de un determinado tipo y es necesario delimitar esta tipología de campos. Para ello, contando con la experiencia de Endalia, se analizan los procesos de recursos humanos más habituales en las compañías y se determinan los siguientes tipos de datos:

- Tipos básicos: entero, decimal, texto, fecha y booleano.
- Tipos complejos: dni y cuenta corriente. Deben validarse para evitar valores incorrectos.
- Tipo fichero. Un caso especial de fichero son las imágenes porque el GUI mostrará su contenido.
- Tipo lista de valores. El usuario debe seleccionar 1 o varios valores de una determinada lista. Algunos casos especiales de este tipo son “Lista de empleados” y “Lista de puestos”. Se han añadido para facilitar el modelado de workflows, ya que este tipo de datos se utiliza en muchos procesos de RRHH.
- Resultado de una consulta. Permite mostrar al usuario información obtenida a través de una consulta al sistema. Este tipo de campos son de sólo lectura.

A partir de este análisis se detectan nuevas funcionalidades para el sistema que se añaden al documento de requisitos:

- Los campos tipo lista de valores pueden variar según el valor de otro campo.
  Un ejemplo de esta funcionalidad sería un formulario que pidiera rellenar ciertos datos del domicilio, como son la provincia y el municipio. El formulario ofreceria para ambos campos una lista de valores para que seleccionara uno. En este caso el valor seleccionado en la provincia condiciona la lista de valores a mostrar en el campo municipio.
- El valor de un campo puede influir en la visibilidad de otro.
  Usando el ejemplo anterior, si el formulario mostrase el campo “País” como una lista de valores, el valor seleccionado (si es “España” o no) condiciona que los campos “Provincia” y “Municipio” se muestren o no.
- Por norma general, los campos transfieren su valor de un paso al siguiente. Cuando un campo es nuevo (no se ha transferido del paso anterior) su valor aparece vacío hasta que el usuario lo rellena.
- Para otorgar mayor flexibilidad al sistema, además de los tipos de campos anteriormente definidos, el sistema permitirá expandirse mediante tipos particulares o tipos *ad-hoc*. Estos campos cumplirán el mismo interfaz que el resto de campos y que permiten leer y guardar su valor, así como representarlo en el GUI. Gracias a ellos, un desarrollador podrá implementar cualquier campo que el cliente necesite y no pueda modelarse con los tipos estándares.

3.6.1 Diagrama de caso de uso

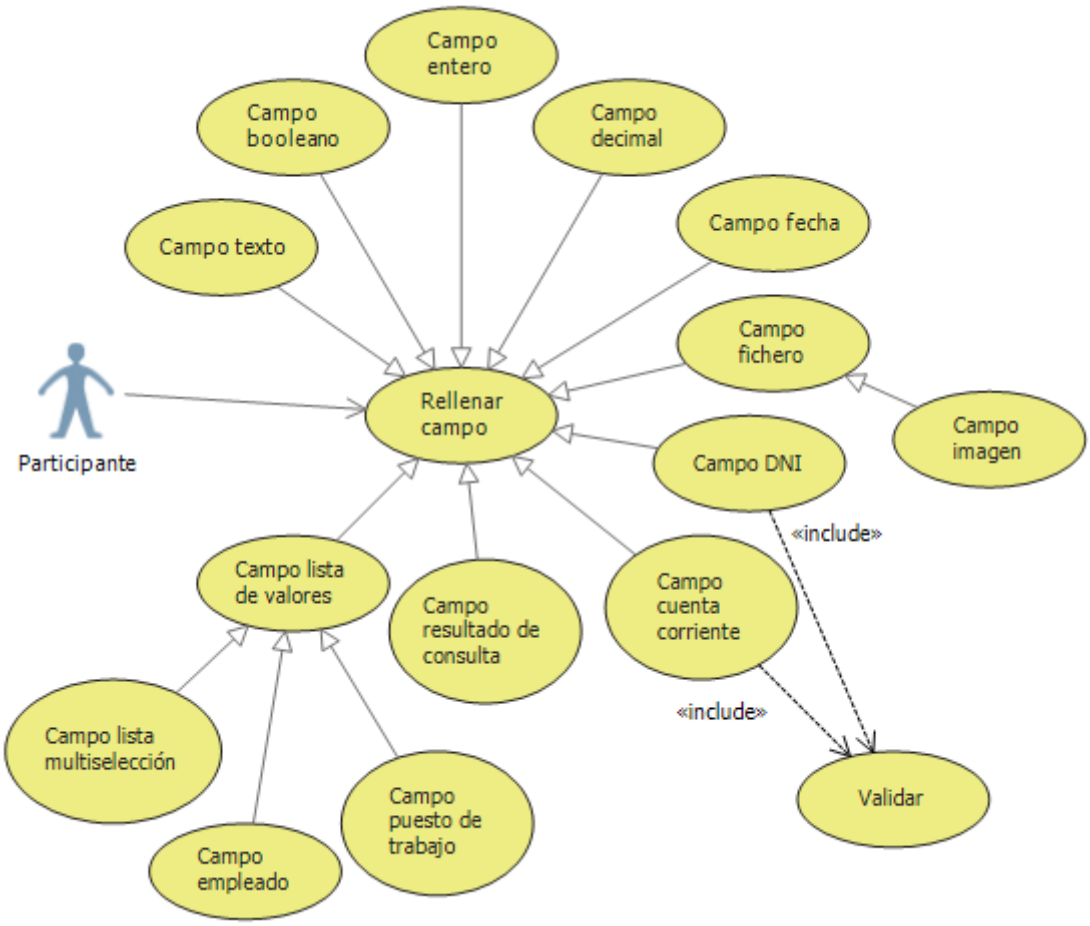


Figura 30: Diagrama de caso de uso Rellenar campo

3.6.2 Caso de uso

Caso de uso:	Rellenar campo
Descripción:	El responsable de un paso rellena un campo de información.
Actores:	El participante del paso en ejecución.
Pre condiciones:	El paso está activo. Tiene campos que no son de sólo lectura.
Flujo normal:	1. El usuario rellena o modifica la información del campo. 2. El sistema valida que la información del campo es correcta. 3. El sistema persiste el valor cambiado. 4. El sistema recalcula el resto de campos del paso. 5. El sistema muestra al usuario la información actualizada.

Flujo alternativo:	Si el usuario no completa correctamente la información del campo, el sistema le muestra un aviso.
Post condiciones:	-

3.6.3 Diagrama de actividad

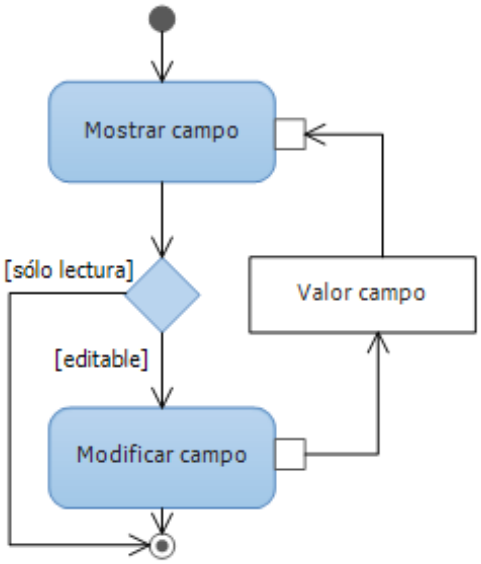


Figura 31: Diagrama de actividad de Rellenar campo

3.6.4 Diagrama robustness

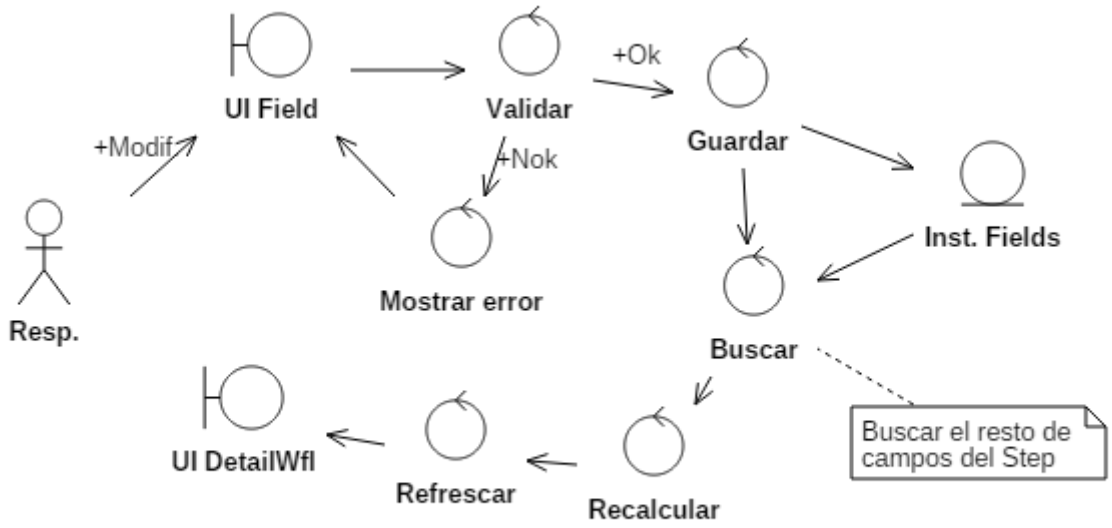


Figura 32: Diagrama robustness Rellenar campo

3.6.5 Diagrama de secuencia

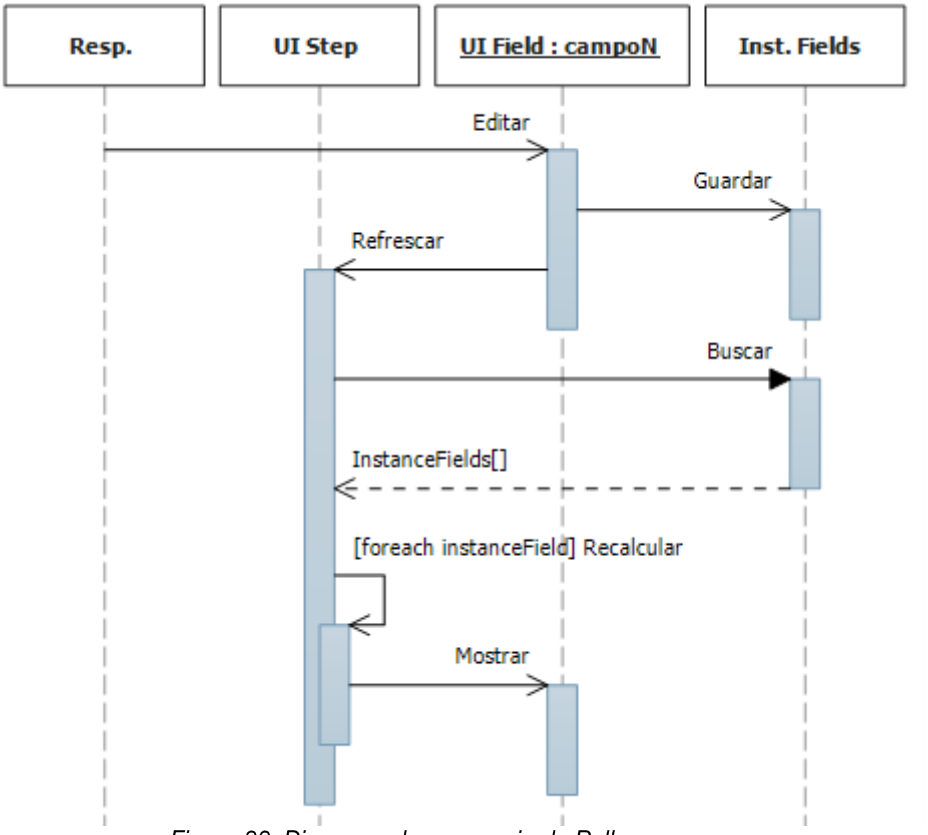


Figura 33: Diagrama de secuencia de Rellenar campo

3.6.6 Diagrama de clases

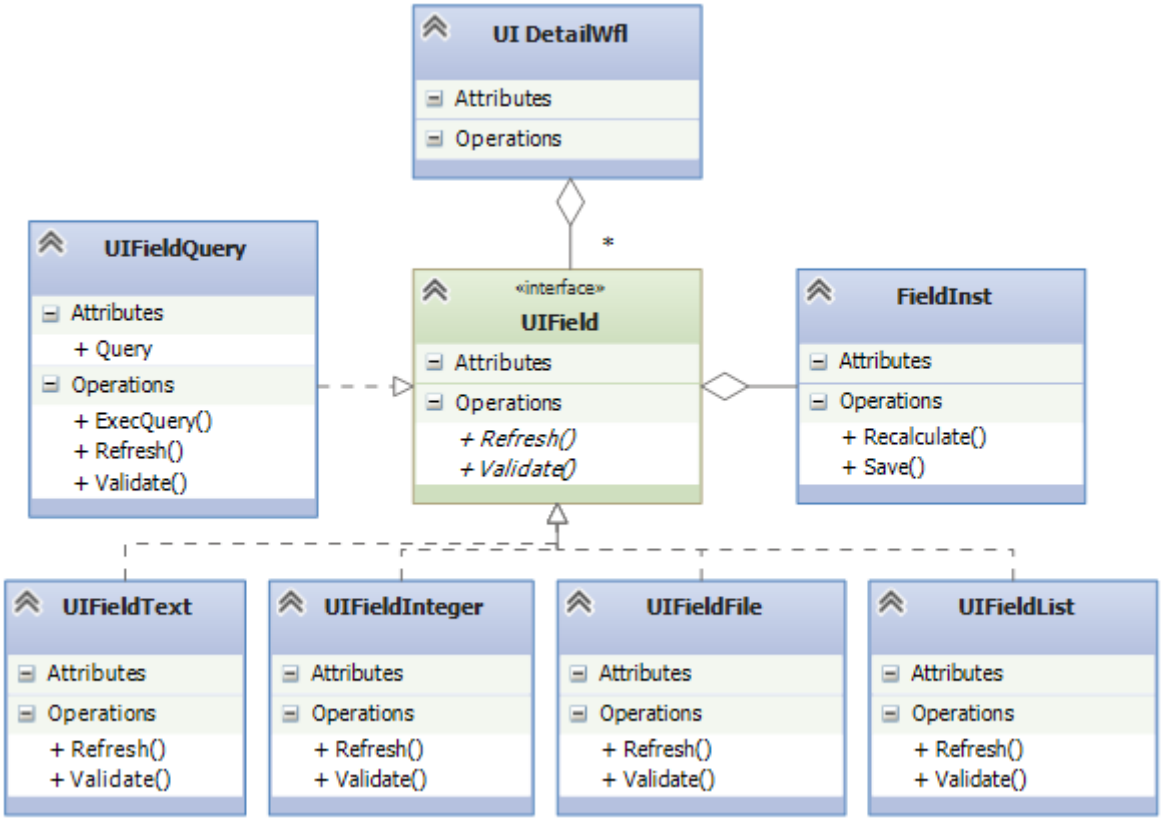


Figura 34: Diagrama de clases caso de uso Rellenar campo

4. DISEÑO DE LA ARQUITECTURA

Atendiendo al análisis de la sección anterior y a los requisitos no funcionales, se diseña una arquitectura de 3 capas para el sistema:

- Capa de presentación o *User Interface* (UI).
- Capa de lógica de negocio o *Business Logic Layer* (BLL).
- Capa de acceso a datos o *Data Access Layer* (DAL).

Para la capa de presentación se ha elegido la tecnología *WebForms* sobre otras opciones como MVC, SPA o *Silverlight*, dado el alto grado de conocimiento del equipo de Endalia en esta tecnología.

Para la persistencia de los datos se usará una base de datos relacional (RDBMS).

Para la capa de acceso a Datos se implementará el patrón de diseño DAO [3] y se utilizará el ORM**¡Error! Marcador no definido.**, *NHibernate*, para convertir las entidades del dominio, que son orientadas a objetos, en entidades relacionales de la Base de datos.

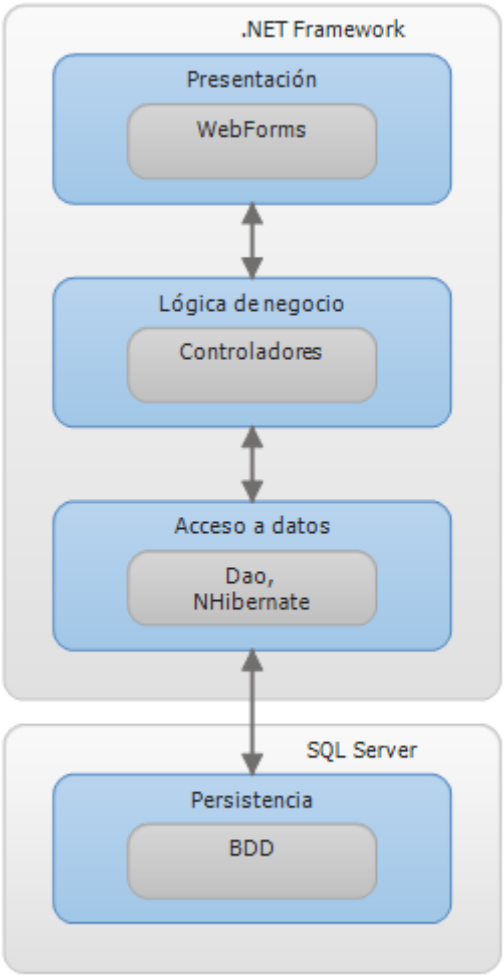


Figura 35: Esquema de la arquitectura del sistema

5. DISEÑO DE LA BASE DE DATOS

A continuación, se diseña el modelo de persistencia que permitirá que la estructura de clases definida en la capa de negocio (modelo de dominio) se persista en base de datos, en este caso, una base de datos relacional. Cabe destacar que para el nombrado de campos y tablas se han seguido las directrices marcadas en el anexo "Estándar de codificación".

5.1 Definición de tablas

Aunque todas las entidades tienen un identificador numérico, en algunas de ellas (Area, Step, Field) se ha añadido un campo "Code" que es un identificador de tipo cadena y que sirve de ayuda al administrador cuando está realizando manipulación de datos.

5.1.1 Orh_Wfl_Workflow y Orh_Wfl_WorkFlowAreas

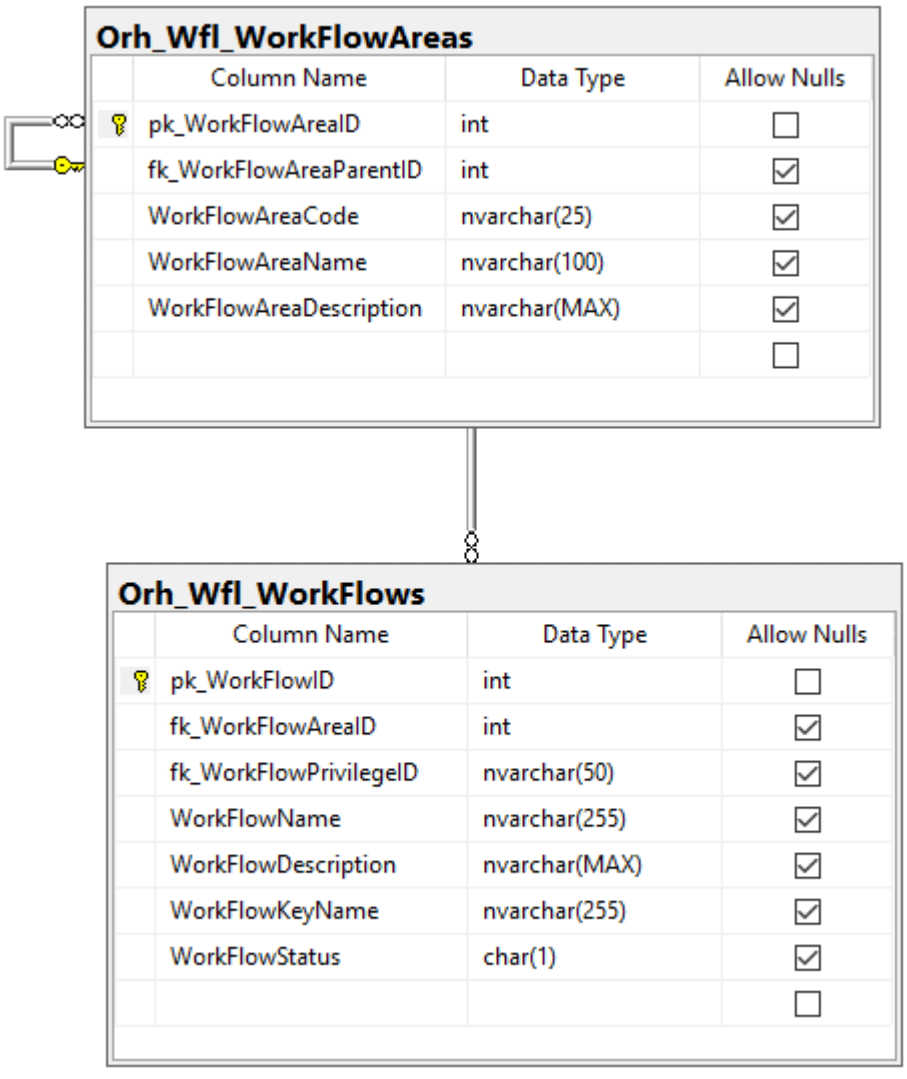
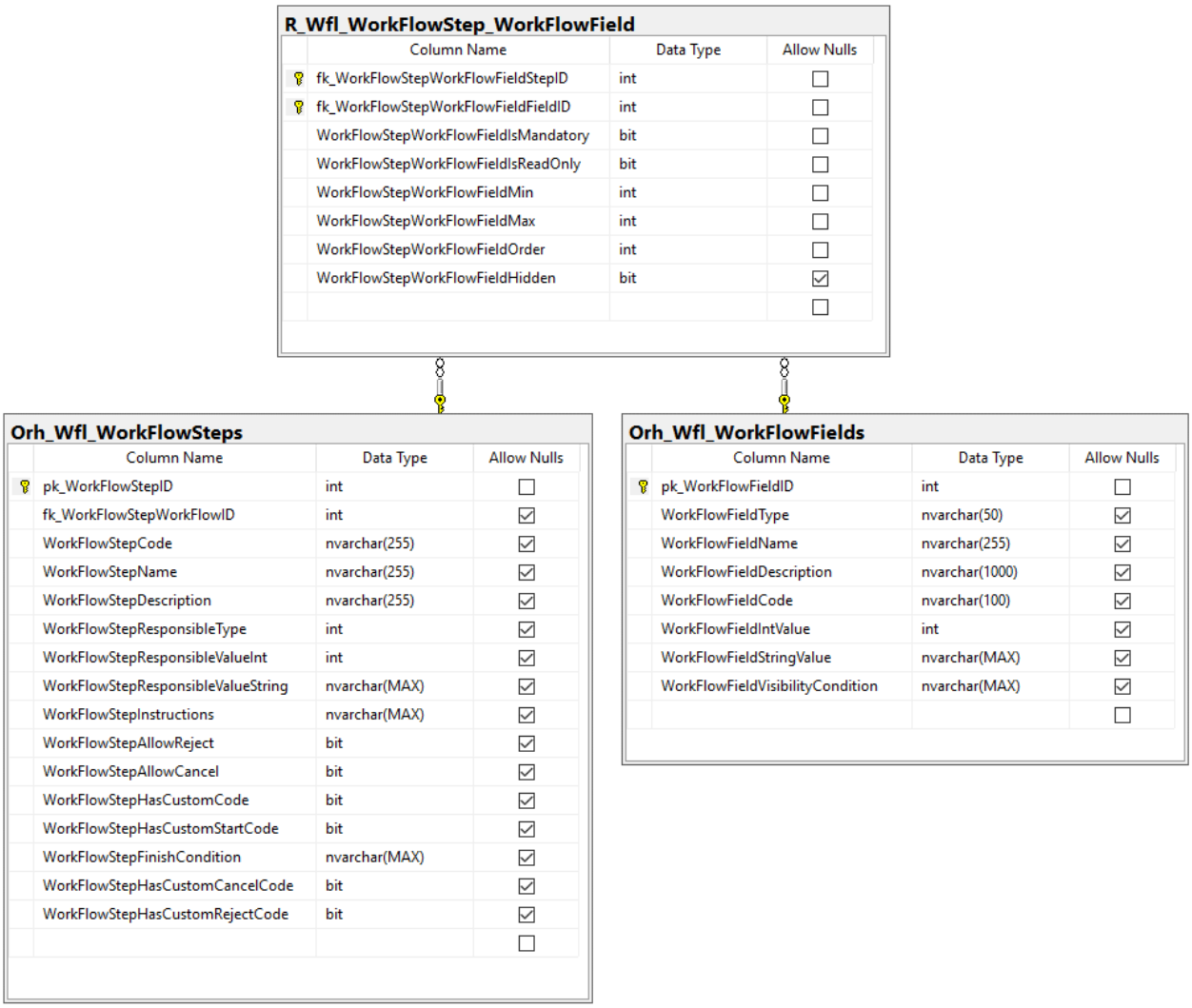


Figura 36: Tabla Orh_Wfl_Workflow

5.1.2 Orh_Wfl_WorkflowStep y Orh_Wfl_WorkflowField



R_Wfl_WorkFlowStep_WorkFlowField

Column Name	Data Type	Allow Nulls
fk_WorkFlowStepWorkFlowFieldStepID	int	☐
fk_WorkFlowStepWorkFlowFieldFieldID	int	☐
WorkFlowStepWorkFlowFieldIsMandatory	bit	☐
WorkFlowStepWorkFlowFieldIsReadOnly	bit	☐
WorkFlowStepWorkFlowFieldMin	int	☐
WorkFlowStepWorkFlowFieldMax	int	☐
WorkFlowStepWorkFlowFieldOrder	int	☐
WorkFlowStepWorkFlowFieldHidden	bit	☑
		☐

Orh_Wfl_WorkFlowSteps

Column Name	Data Type	Allow Nulls
pk_WorkFlowStepID	int	☐
fk_WorkFlowStepWorkFlowID	int	☑
WorkFlowStepCode	nvarchar(255)	☑
WorkFlowStepName	nvarchar(255)	☑
WorkFlowStepDescription	nvarchar(255)	☑
WorkFlowStepResponsibleType	int	☑
WorkFlowStepResponsibleValueInt	int	☑
WorkFlowStepResponsibleValueString	nvarchar(MAX)	☑
WorkFlowStepInstructions	nvarchar(MAX)	☑
WorkFlowStepAllowReject	bit	☑
WorkFlowStepAllowCancel	bit	☑
WorkFlowStepHasCustomCode	bit	☑
WorkFlowStepHasCustomStartCode	bit	☑
WorkFlowStepFinishCondition	nvarchar(MAX)	☑
WorkFlowStepHasCustomCancelCode	bit	☑
WorkFlowStepHasCustomRejectCode	bit	☑
		☐

Orh_Wfl_WorkFlowFields

Column Name	Data Type	Allow Nulls
pk_WorkFlowFieldID	int	☐
WorkFlowFieldType	nvarchar(50)	☑
WorkFlowFieldName	nvarchar(255)	☑
WorkFlowFieldDescription	nvarchar(1000)	☑
WorkFlowFieldCode	nvarchar(100)	☑
WorkFlowFieldIntValue	int	☑
WorkFlowFieldStringValue	nvarchar(MAX)	☑
WorkFlowFieldVisibilityCondition	nvarchar(MAX)	☑
		☐

Figura 37: Tabla Orh_Wfl_WorkflowStep

5.1.3 Orh_Wfl_WorkflowInstanceField

Los valores de los campos se han separado en 2 tablas por motivos de eficiencia, así los valores de tipo fichero (pueden ser valores muy grandes de tamaño) se almacenan en una tabla distinta.

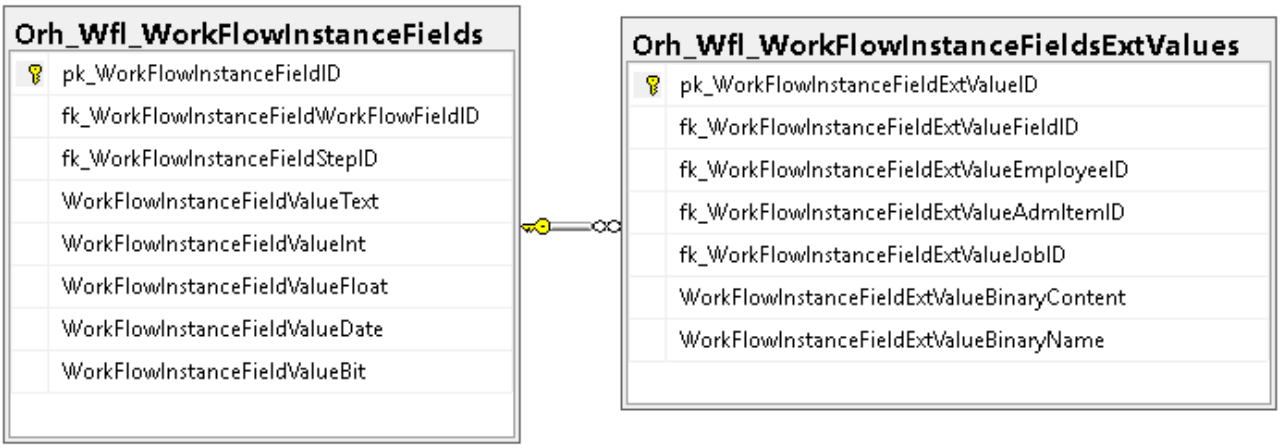


Figura 38: Tablas InstanceFields

5.2 Modelo relacional

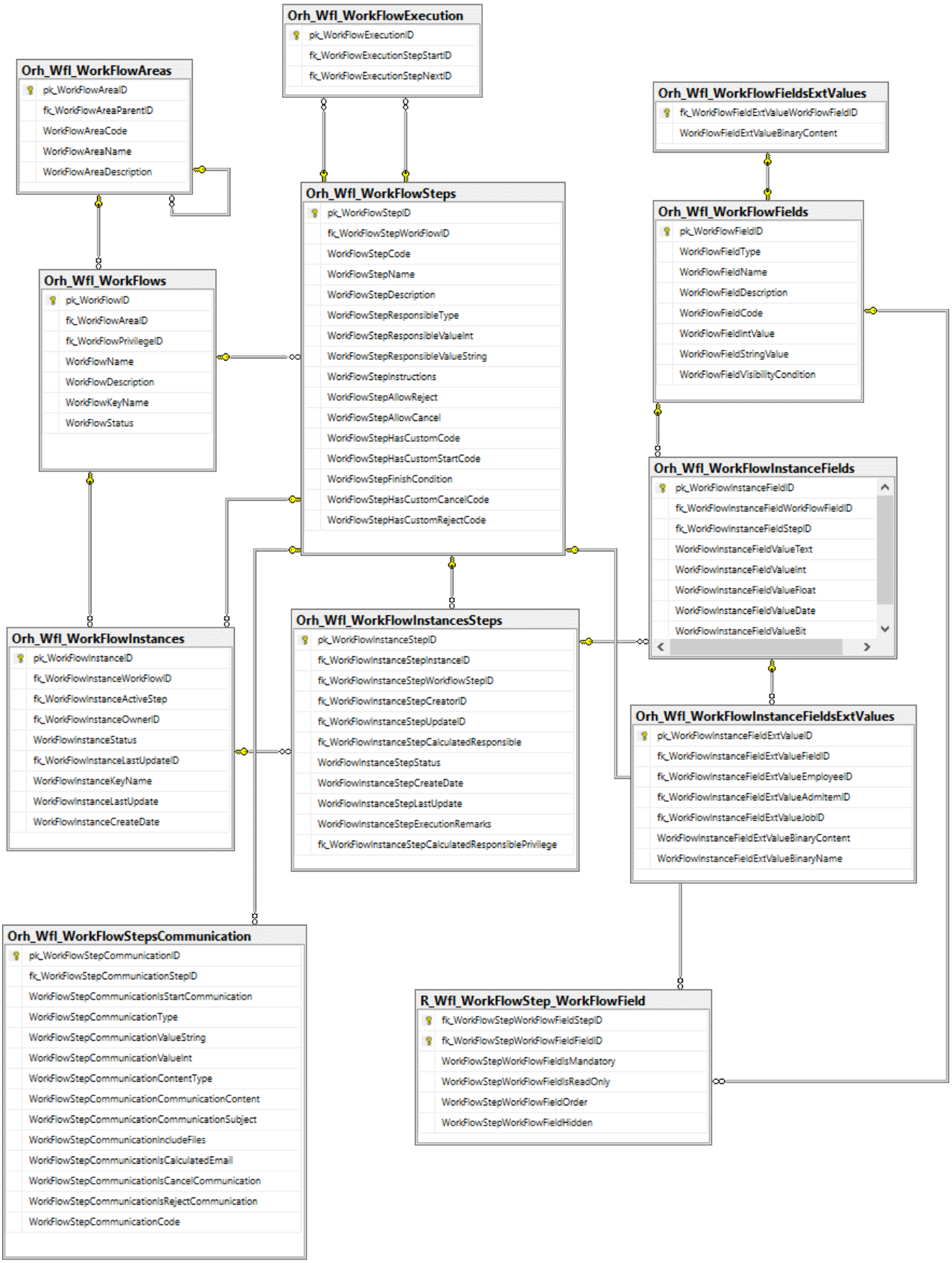


Figura 39: Modelo de base de datos

6. BIBLIOGRAFÍA

[1] «OOAD,» [En línea]. Available: https://en.wikipedia.org/wiki/Object-oriented_analysis_and_design.

[2] «Anexo I. Documento Visión, sección 2.3».

[3] Wikipedia, «DAO pattern,» [En línea]. Available: https://es.wikipedia.org/wiki/Data_Access_Object.

[4] Doug Rosenberg, Kendal Scott, Applying Use Case Driven Object Modelling with UML, Addison-Wesley, ISBN 0-201-73039-1, 2001.

endalia®
HR Consulting & Software

VIII. MANUAL DE USUARIO

WORKFLOW MANAGEMENT SYSTEM

DAVID ROY

TABLA DE CONTENIDOS

Tabla de contenidos 2
1. Introducción 3
1.1 Objetivo 3
1.2 Alcance 3
1.3 Organización del documento 3
2. EndaliaWeb 4
2.1 Pantalla de acceso 4
2.2 Pantalla de inicio 4
2.3 Menú lateral 5
2.4 Breadcrums 5
2.5 Tablas 5
2.5.1 Movimiento de columnas 6
2.5.2 Ordenación de filas 6
2.5.3 Agrupación de filas 6
3. Módulo de workflows 7
3.1 Menú 7
3.2 Mis workflows 7
3.3 Iniciar nueva instancia 8
3.4 Pantalla de workflow 9
3.5 Campos de formulario 10
3.5.1 Fecha 10
3.5.2 Fichero 10
3.5.3 Lista 11
3.5.4 Empleado 11
3.5.5 Cuenta corriente 12
3.6 Documentos del módulo 12
4. Bibliografía 13

1. INTRODUCCIÓN

1.1 Objetivo

El objetivo de este documento es presentar una guía rápida de uso del sistema, a modo de *Getting Started Guide*, orientada a los usuarios que inician por primera vez la aplicación.

1.2 Alcance

La filosofía de Endalia para la documentación de ayuda al usuario se basa en la elaboración de guías rápidas con capturas de pantalla y breves descripciones de la funcionalidad más relevante de dicha pantalla.

1.3 Organización del documento

El presente documento contiene las siguientes secciones:

1. Introducción: Descripción del objetivo y alcance del documento.
2. EndaliaWeb: Descripción de los elementos genéricos de la aplicación EndaliaWeb.
3. Módulo Workflows: Descripción de los componentes particulares del módulo de workflows.
4. Bibliografía: Referencias a libros y webs utilizadas en el documento.

2. ENDALIAWEB

En esta sección se van a explicar los elementos fijos de la aplicación EndaliaWeb, así como los controles genéricos.

2.1 Pantalla de acceso

Para iniciar la aplicación debemos autentificarnos con nuestro usuario y contraseña a través de la pantalla de acceso:

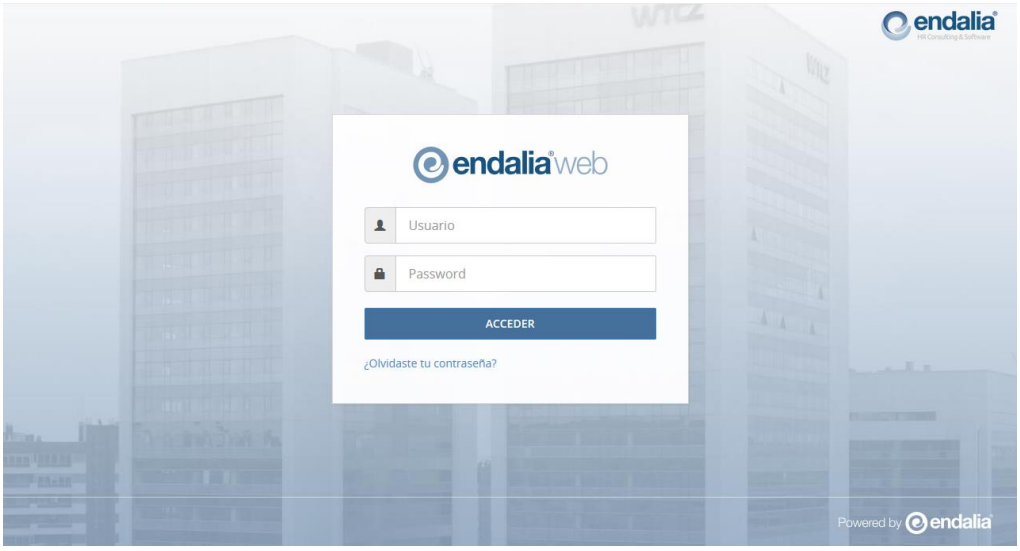


Figura 1: Pantalla de acceso

2.2 Pantalla de inicio

Una vez nos hemos autentificado en el sistema, aterrizamos en nuestra pantalla de inicio:

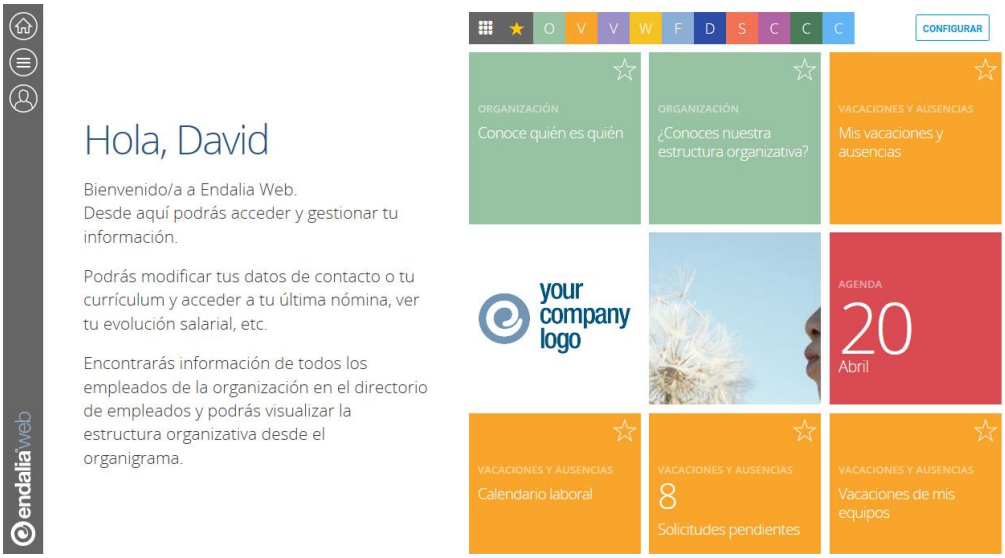


Figura 2: Pantalla de inicio

En esta pantalla tenemos varios elementos:

- Menú lateral: nos permite navegar por todas las funcionalidades de la aplicación.
- Texto de bienvenida: introducción amigable que nos explica las principales funcionalidades.
- Botones de acceso rápido: nos permiten acceder directamente a las pantallas más habituales.

2.3 Menú lateral

El menú lateral está situado en la parte izquierda de la pantalla y está siempre visible. Se distingue del resto de contenidos de la pantalla por su color más oscuro.

En la parte superior del menú se encuentran 3 botones:

Figura 3: Menú lateral

De arriba abajo, estos botones se corresponden con:

- Inicio: al pulsarlo vamos directamente a la pantalla de inicio.
- Desplegar menú: al pulsarlo podremos navegar por los diferentes menús de la aplicación.
- Opciones de usuario: nos da acceso a las opciones de configuración, así como al botón de salida de la aplicación.

2.4 Breadcrums

Puedes conocer en todo momento en qué pantalla te encuentras y navegar a las anteriores gracias al *breadcrum* o miga de pan [1] que se encuentra en la parte superior de la pantalla, justo debajo del título:

Figura 4: Ejemplo de breadcrum

2.5 Tablas

Las tablas que se muestran en la aplicación son configurables y permiten realizar estas acciones:

- Movimiento de columnas.
- Ordenación de filas.
- Agrupación de filas.

2.5.1 Movimiento de columnas

Para mover una columna de la tabla a otra posición sólo tiene que pinchar sobre la cabecera de columna y arrastrarla a la nueva posición deseada.

2.5.2 Ordenación de filas

Puede ordenar las filas de una tabla en base a los datos de una columna. Para ello sólo tiene que pinchar sobre la cabecera de la columna y los datos se ordenarán de forma creciente. Si vuelve a pinchar sobre la columna, los datos se ordenarán de forma decreciente.

Recuerde que puede ordenar por varias columnas a la vez, repitiendo los pasos de arriba en cada una de las columnas.

2.5.3 Agrupación de filas

En la parte superior de las tablas, justo encima de las cabeceras de columna, aparece la barra de agrupación:

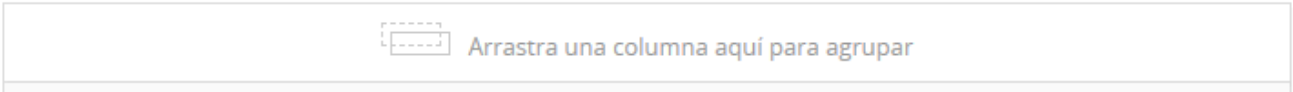


Figura 5: Barra de agrupación

Si pincha sobre una cabecera de columna y la arrastra a esta barra, los datos de la tabla se agruparán conforme a los valores de dicha columna. Los grupos de filas pueden expandirse y contraerse para mayor comodidad.

Las columnas agrupadas aparecen en la barra de agrupación de la siguiente manera:

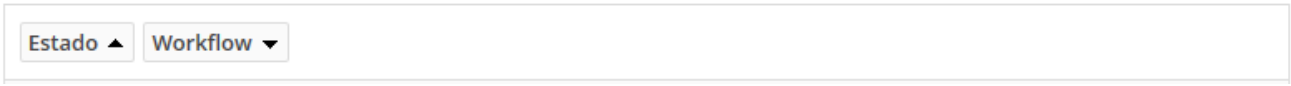


Figura 6: Ejemplo de columnas agrupadas

Las columnas agrupadas también permiten ordenación de filas, del mismo modo que las columnas no agrupadas, tan sólo con pinchar encima de ellas.

Para eliminar una agrupación ponga el cursor encima de la cabecera de columna que desea desagrupar y aparecerá el icono de eliminación, representado por círculo con un aspa:

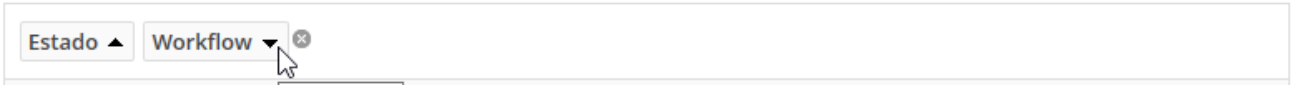


Figura 7: Botón de eliminar agrupación

3. MÓDULO DE WORKFLOWS

3.1 Menú

El módulo de workflows está disponible desde el menú lateral y tiene las siguientes secciones:

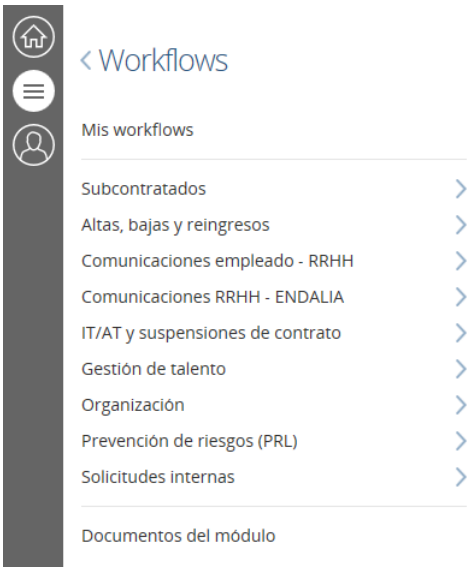


Figura 8: Menú de workflows

La primera sección del menú muestra el acceso a la pantalla de “Mis workflows” desde la que podrá consultar sus workflows y en qué estado se encuentran (en el siguiente apartado se explica más en detalle esta pantalla).

A continuación, se muestran las áreas de workflows configuradas en el sistema y permiten navegar por los diferentes workflows de cada una de ellas. Pinchando en cualquiera de los workflows se inicia una nueva instancia.

Por último, aparece la sección de acceso a los documentos del módulo, donde podrá encontrar documentación y ficheros de ayuda de los worfklows, entre otros, la presente guía de uso.

3.2 Mis workflows

La pantalla “Mis workflows” le permite consultar sus workflows y realizar diferentes tipos de filtrados:

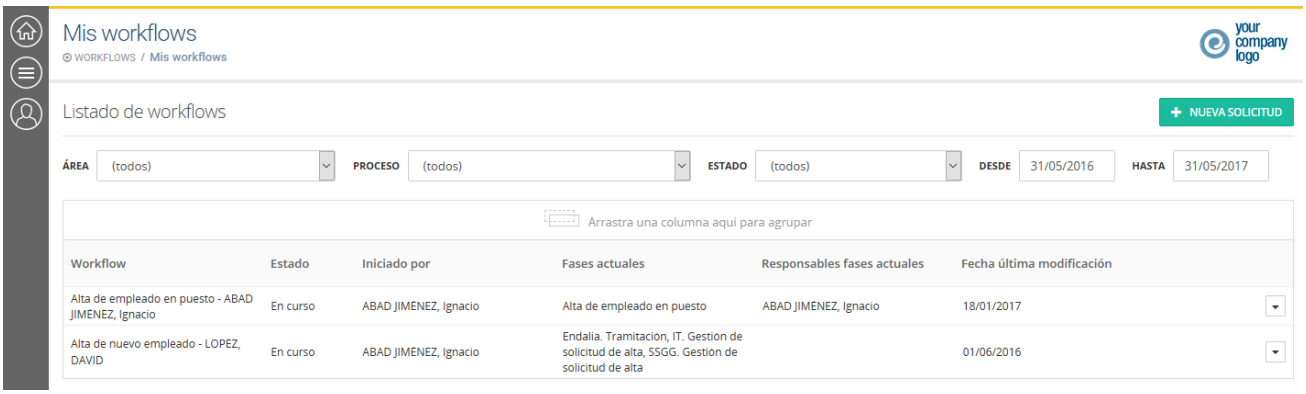


Figura 9: Pantalla "Mis workflows"

En la parte superior derecha de la pantalla se encuentra el botón para iniciar una nueva instancia del workflow (en el siguiente apartado se explica en detalle).

Justo debajo se muestran los campos de filtrado y la tabla de resultados. En cuanto Ud. modifica alguno de los filtros, la tabla con los resultados se actualiza automáticamente.

Recuerde que, tal y como se ha explicado en el apartado 2.5 de este documento, puede realizar diferentes operaciones en la tabla de resultados, tales como ordenar o agrupar filas.

En cada fila de resultados, a la derecha, se encuentra el botón de opciones de instancia. Si lo pulsa podrá acceder a ellas:

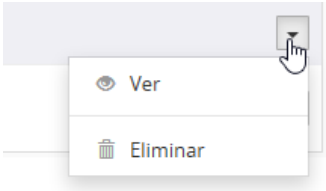


Figura 10: Opciones de instancia

3.3 Iniciar nueva instancia

Para iniciar una nueva instancia de workflow puede hacerlo bien desde el menú lateral, accediendo al área y luego al workflow que desea iniciar. O también puede pulsar el botón de “Nueva solicitud” de la pantalla de “Mis workflows”. En este último caso, deberá indicar en el popup el tipo de workflow que desea iniciar:

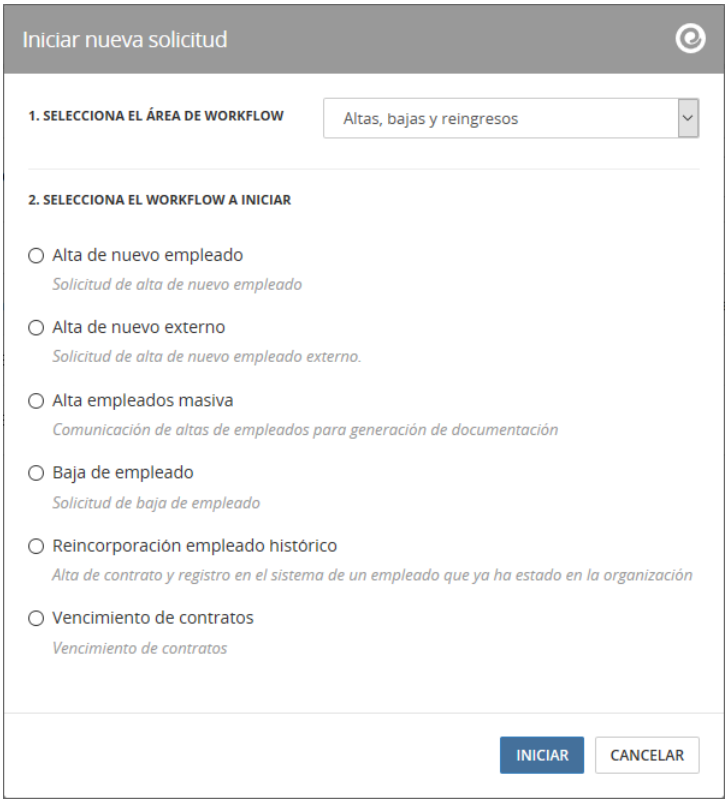


Figura 11: Popup iniciar nueva instancia

3.4 Pantalla de workflow

Una vez ha iniciado una nueva instancia, o cuando accede a una ya existente, se muestra una pantalla con una estructura similar a la siguiente:

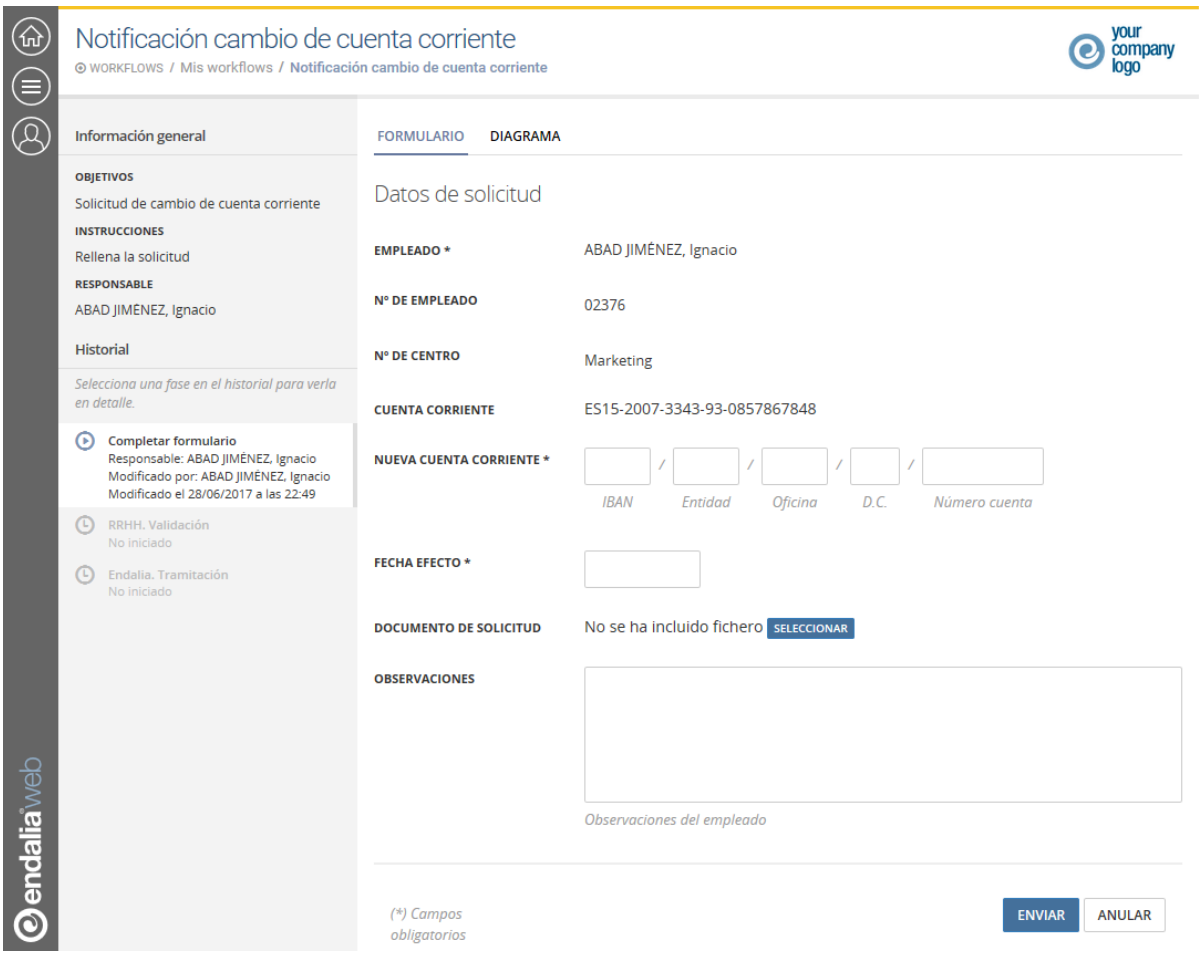


Figura 12: Ejemplo de pantalla de un paso de workflow

En la parte izquierda de la pantalla se muestra información acerca del proceso, los responsables y fechas.

La parte central se encuentra dividida en 2 pestañas:

- Formulario
- Diagrama

El formulario muestra los campos del paso que está en curso, para que Ud. pueda consultarlos y rellenarlos. Note que algunos de estos campos pueden ser de sólo lectura y no podrá modificarlos.

En la parte de abajo están disponibles los botones de acción que le permiten avanzar el workflow (enviar el formulario), anularlo y en algunos casos rechazarlo al paso o pasos anteriores.

Por su parte la pestaña “Diagrama” muestra un esquema de los diferentes pasos que componen el workflow:

Figura 13: Diagrama de pasos de un workflow

3.5 Campos de formulario

Existen diferentes tipos de campos de formulario, a continuación, vamos a explicar las características de los más destacados.

3.5.1 Fecha

En los campos tipo fecha, Ud. puede introducir la fecha directamente a través del teclado o puede utilizar el calendario que se muestra para seleccionar cómodamente el día:

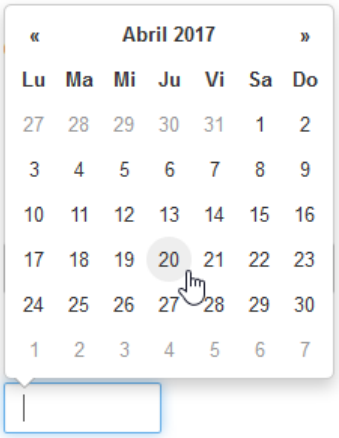


Figura 14: Ejemplo de campo tipo fecha

3.5.2 Fichero

Los campos tipo fichero le permiten subir un fichero nuevo al workflow:

Figura 15: Popup para subir fichero a un campo

Y también le permiten descargar un fichero que haya sido subido anteriormente al workflow:

Figura 16: Ejemplo descarga de campo fichero

3.5.3 Lista

Este tipo de campos muestra una lista de valores y Ud. debe seleccionar uno de ellos:

Figura 17: Campo tipo lista

3.5.4 Empleado

Este tipo de campos se utilizan para seleccionar uno o varios empleados de la compañía. Los empleados se muestran en un *popup*, organizados en árbol según su puesto laboral:

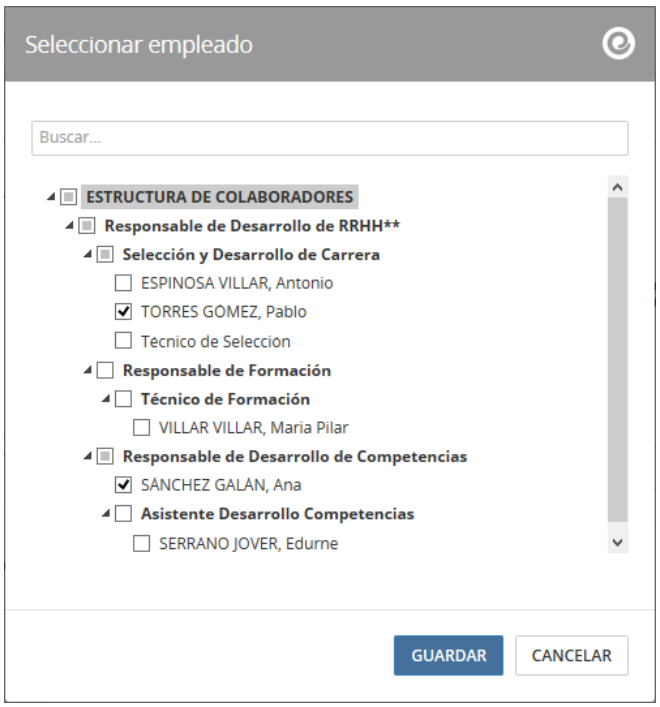


Figura 18: Campo de tipo empleado

3.5.5 Cuenta corriente

Este tipo de campos se utilizan para almacenar un nº de cuenta y por ello presentan este aspecto:

Figura 19: Campo de cuenta corriente

El valor introducido se verifica y en caso de que no se corresponda con un nº de cuenta corriente válido, se advierte al usuario y se impide continuar:

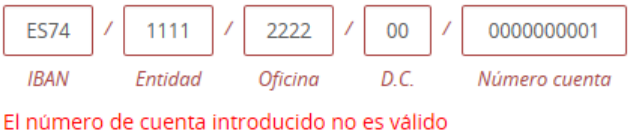


Figura 20: Campo de cuenta corriente incorrecto

3.6 Documentos del módulo

Desde esta pantalla tendrá acceso a diferentes documentos que pueden resultar de utilidad para el uso del módulo de workflows. Los ficheros están organizados en directorios y subdirectorios:

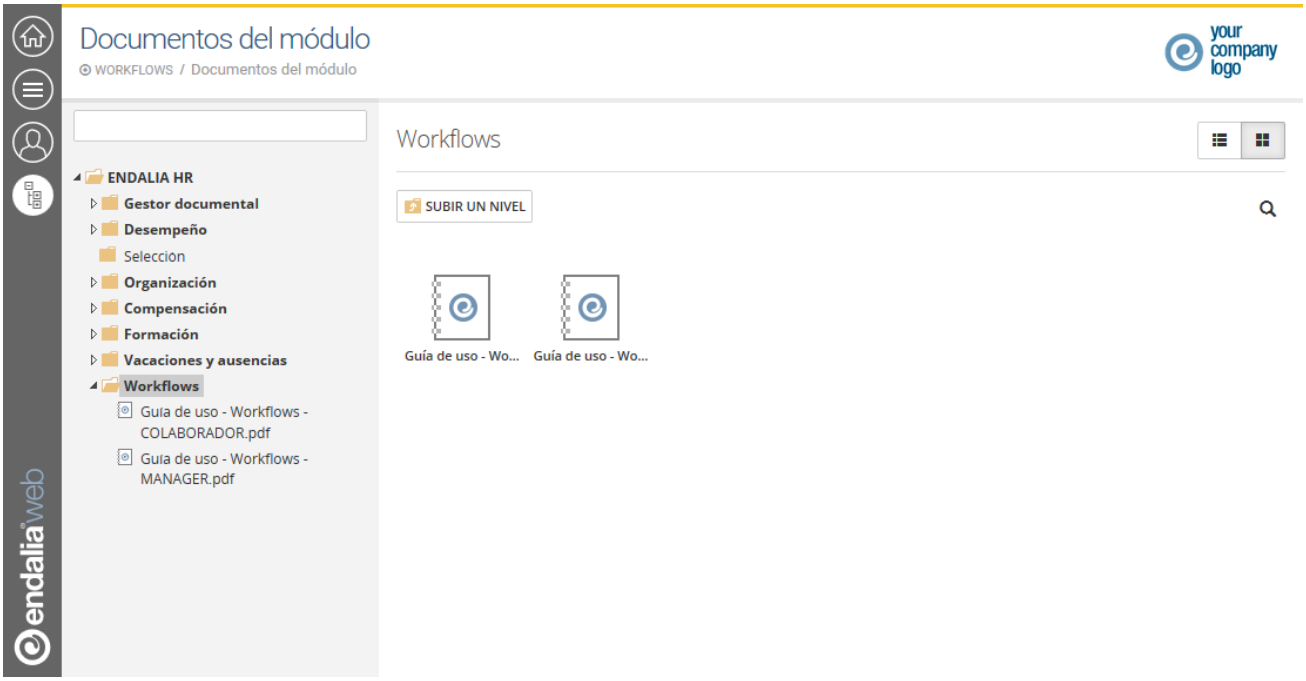


Figura 21: Pantalla de documentos del módulo

4. BIBLIOGRAFÍA

[1] Wikipedia, «Breadcrum,» [En línea]. Available: https://es.wikipedia.org/wiki/Miga_de_pan_(inform%C3%A1tica).

endalia®
HR Consulting & Software

IX. PLAN DE PRUEBAS

WORKFLOW MANAGEMENT SYSTEM

DAVID ROY

TABLA DE CONTENIDOS

Tabla de contenidos 2
1. Introducción 3
1.1 Objetivo 3
1.2 Alcance 3
1.3 Organización del documento 3
2. Descripción del proceso 3
3. Pruebas de integración 4
3.1 Pruebas de integración 4
4. Pruebas de Sistema y de Aceptación 8
4.1 Pruebas de operación 8
4.2 Pruebas de facilidad de uso 8

1. INTRODUCCIÓN

1.1 Objetivo

El objetivo de este documento se centra en certificar y asegurar la calidad del software.

1.2 Alcance

El alcance de este documento es realizar las pruebas al software que se está desarrollando desde las primeras fases de desarrollo hasta la fase final de despliegue.

1.3 Organización del documento

El presente documento contiene las siguientes secciones:

1. Introducción: Descripción del objetivo y alcance del documento.
2. Descripción del proceso.
3. Pruebas de sistema.
4. Pruebas de aceptación.

2. DESCRIPCIÓN DEL PROCESO

En programación, una prueba unitaria es una forma de comprobar el correcto funcionamiento de una unidad de código. Esto sirve para asegurar que cada unidad funcione correctamente y eficientemente por separado. Además de verificar que el código hace lo que tiene que hacer, se verifica el número y tipo de los parámetros y valores devueltos.

Las pruebas unitarias no deben conectar con otros sistemas (incluida la base de datos) y en su lugar utilizan *stubs* (objetos que tienen un comportamiento programado ante ciertas llamadas de un test en concreto) o *mocks (objetos ya programados con los datos que se espera recibir)* que simulan el otro sistema.

Definición:

- Prueba unitaria: prueba un único método de una clase. El alcance es muy reducido y está perfectamente acotado. Cualquier dependencia del módulo bajo prueba debe ser sustituida por un *mock*, o un *stub*.
- Prueba de integración: prueba la interacción entre dos o más elementos, que pueden ser clases, módulos, paquetes, subsistemas, etc… incluso la interacción del sistema con el entorno de producción.

A continuación, se definen pruebas de integración para cada funcionalidad no trivial, con el objetivo de poder asegurar el correcto funcionamiento del sistema.

3. PRUEBAS DE INTEGRACIÓN

3.1 Pruebas de integración

Dado que sólo tenemos una interfaz para acceder a la aplicación, se debe implementar un conjunto de pruebas de integración (en función de los valores de entrada y condiciones iniciales) por cada método de esta interfaz.

Para facilitar las pruebas, se ha definido un tipo de workflow básico, con 2 fases y 1 campo de cada tipo en ellas. Para facilitar las pruebas, este workflow tiene configuradas 3 notificaciones: avance, retroceso o cancelación en las que el asunto del email indica de qué notificación se trata. Además, tiene definidas acciones diferenciadas para avanzar y para retroceder, que simplemente dejan traza en el fichero de *log*.

Cuando las pruebas requieran workflows diferentes, se copiará el workflow de ejemplo pero con un nombre diferente. Así a efectos funcionales será un workflow diferente.

Identificador	PI-01
Descripción	Comprobar la consulta de workflows filtrado por estado
Método	Preparar datos con instancias de por lo menos 2 workflows diferentes. Poner instancias en cada uno de los estados: Activo, Finalizado, Cancelado y No iniciado.
Criterio aceptación	Devuelve en cada caso las instancias correspondientes al filtro.
Resultado	Ok

Identificador	PI -02
Descripción	Comprobar la consulta de workflows filtrado por fecha
Método	Preparar datos con instancias de por lo menos 2 workflows diferentes. Poner fechas de última modificación diferente en cada instancia.
Casos especiales	Comprobar que la fecha inicial y final del filtro están incluidas en los resultados
Criterio aceptación	Devuelve en cada caso las instancias correspondientes al rango de fechas del filtro.
Resultado	Ok

Identificador	PI -03
Descripción	Comprobar la consulta de workflows filtrado por autor
Método	Preparar datos con instancias de por lo menos 2 workflows diferentes. Poner autores diferentes.
Criterio aceptación	Devuelve en cada caso las instancias correspondientes al filtro.
Resultado	Ok

Identificador	PI -04
Descripción	Comprobar la consulta de workflows filtrados por nombre de workflow
Método	Preparar datos con instancias de por lo menos 2 workflows diferentes.
Criterio aceptación	Devuelve en cada caso las instancias correspondientes al filtro.
Resultado	Ok

Identificador	PI -05
Descripción	Comprobar la consulta de workflows filtrado por áreas
Método	Preparar datos con instancias de por lo menos 2 workflows diferentes de 2 áreas diferentes.
Criterio aceptación	Devuelve en cada caso las instancias correspondientes al filtro.
Resultado	Ok

Identificador	PI -06
Descripción	Comprobar la consulta de workflows filtrado por varios criterios a la vez
Método	Preparar 4 instancias de por lo menos 2 workflows diferentes. Poner instancias en cada uno de los estados: Activo, Finalizado, Cancelado y No iniciado, fechas diferentes y por lo menos 2 autores diferentes.

Criterio aceptación	Devuelve en cada caso las instancias correspondientes a los filtros.
Resultado	Ok

Identificador	PI -07
Descripción	Comprobar envío de emails al avanzar/retroceder/cancelar una instancia
Método	Iniciar una instancia de workflow de al menos 2 pasos, que tenga en el paso 1 habilitada la cancelación y en el paso 2 habilitado el rechazo. Configurar los correos de avance, rechazo y cancelación. Avanzar del paso 1 al paso 2 y comprobar correo. Rechazar paso 2 y volver al paso 1, comprobar correo y finalmente cancelar la instancia y comprobar correo.
Criterio aceptación	En cada una de las acciones ejecutadas (avance, rechazo y cancelación) envía un email de notificación indicando la acción ejecutada.
Resultado	Ok

Identificador	PI -08
Descripción	Comprobar que, al acceder a un workflow sin permiso, no se ve la información.
Método	Preparar instancias de workflow y acceder con un usuario que no sea responsable ni autor.
Criterio aceptación	El usuario no puede ver los campos de información del workflow. NOTA: El usuario puede hacer consultas sobre workflows y en los resultados de esas consultas pueden aparecer workflows que no son de su responsabilidad, pero no podrá ver la información contenida dentro del workflow si no es responsable del paso.
Resultado	Ok

Identificador	PI -09
Descripción	Comprobar campos obligatorios.
Método	Preparar una instancia con varios campos de tipo obligatorio. Intentar avanzar de fase sin rellenar 1 de esos campos

Criterio aceptación	El sistema no permite avanzar de fase y avisa al usuario de que debe completar todos los campos obligatorios y resalta el campo que falta por rellenar. Una vez rellenado el campo sí permite avanzar de fase.
Resultado	Ok

Identificador	PI -10
Descripción	Comprobar campos de sólo lectura.
Método	Preparar una instancia con varios campos de sólo lectura. Intentar editar el valor de ese campo.
Criterio aceptación	El usuario no puede modificar ninguno de los campos de sólo lectura.
Resultado	Ok

Identificador	PI -11
Descripción	Comprobar condición de fin de paso.
Método	Se prepara una instancia con un campo de texto y una condición de fin de paso que compruebe si el campo de texto contiene un determinado valor.
Criterio aceptación	Al avanzar de fase, si el campo de texto no cumple con el valor que hemos indicado en la condición de fin de paso, no permite avanzar y avisa al usuario de tal circunstancia. Una vez rellenado correctamente el valor sí permite avanzar de fase.
Resultado	Ok

Identificador	PI -12
Descripción	Comprobar validación de campos tipo NIF.
Método	Preparar una instancia con un campo de tipo NIF.
Criterio aceptación	Rellenar el campo con un NIF inválido y comprobar que el sistema avisa de tal circunstancia. Una vez rellenado un NIF válido, el sistema quita el aviso.
Resultado	Ok

4. PRUEBAS DE SISTEMA Y DE ACEPTACIÓN

4.1 Pruebas de operación

En estas pruebas se ha determinado el cumplimiento de los procedimientos de inicio y fin de trabajo y el cumplimiento de las especificaciones y requisitos.

4.2 Pruebas de facilidad de uso

En estas pruebas se ha comprobado la adaptabilidad del sistema a las necesidades de los usuarios y al método habitual de trabajo con una herramienta informática común. Se ha verificado asimismo que las interfaces gráficas que componen el sistema, y con las que interactúa el usuario, son adecuadas, sencillas y agradables visualmente.